LOAN PREDICTION BASED ON NATURE OF THE CLIENTS

A Project Report:

Submitted as part of the Course

"Exploratory Data Analysis" (CSE3040)

School of Computer Science and Engineering

VIT Chennai.

(Winter 2020-2021)

Course Faculty : Prof. Subhra Patra

Submitted by:

- Mohammed Imran.Z (19MIA1047)
- Aishwarya.S (19MIA1063)
- Keerthana Madhavan (19MIA1073)
- Podalakuru Sahithya (19MIA1084)

INDEX

Abstract:

In India, the number of people applying for the loans gets increased for various reasons in recent years. The bank employees are not able to analyse or predict whether the customer can payback the amount or not (good customer or bad customer) for the given interest rate. The aim of this paper is to find the nature of the client applying for the personal loan. An exploratory data analysis technique is used to deal with this problem. The primary objective of the bank is to provide their wealth in the safer hands. In recent times, banks approve loan after verifying and validating the documents provided by the customer. Yet there is no guarantee whether the applicant is deserving or not. This paper classifies the customers based on certain criteria. The result of the analysis shows that short term loans are preferred by majority of the clients and the clients majorly apply loans for debt consolidation. The results are shown in graphs that helps the bankers to understand the client's behaviour.

Introduction:

The term Banking can be defined as receiving and protecting money that is deposited by the individual or the entities. This also includes lending money to the people which will be repaid within the given time. Banking sector is regulated in most of the countries as it is the important factor in determining the financial stability of the country.The provision of banking regulation act allows public to obtain loans.Loans are good sum of money borrowed for a period and expected to be paid back at given interest rate. The purpose of the loan can be anything based on the customer requirements. Loans are broadly divided as openended and close-ended loans. Open-ended loans are the loans for which the client has approval for a specific amount.

Examples of open-end loans are credit cards and a home equity line of credit (HELOC). Close-ended loans decreases with each payment. In other words, it is a legal term that cannot be modified by the borrower. Personal loans, mortgages, auto payments, instalment loan and student loans are the most common examples of close-ended loans. Secured or collateral loan are those loans that are protected by an asset. Houses, Vehicles, Savings accounts are the personal properties used to secure the loan.

Unsecured loans are also known as personal or signature loans. Here the lender believes that the borrower can repay the loan based on financial resources possessed by the borrower. Liquidity risk is the risk that arises from the lackof marketability of an investment that cannot be bought or sold quickly enough to prevent or minimize a loss. The interest rate risk is the risk in which the interest rates priced on loans will be too low to earn the bank money.

The primary objective of the bank is to provide their wealth in the safer hands. In recent times, banks approve loan after verifying and validating the documents provided by the customer. Yet there is no guarantee whether the applicant is deserving or not. This paper classifies the customers based on certain criteria. The classification is done using Exploratory Data Analysis. Exploratory Data Analysis (EDA) is an approach to analyse the datasets that summarizes the main characteristics with visual methods. The purpose of using EDA is to uncover the underlying structure of a relatively larger set of variables using visualizing techniques.

Literature Survey / Related Works :

In [1] the researchers analyse the data set using data mining technique. Data mining procedure provides a great vision in loan prediction systems, since this will promptly distinguish the customers who are able to repay the loan amount within a period. Algorithms like “J48 algorithm”, “Bayes net”, Naive Bayes” are used. On applying these algorithms to the datasets, it was shown that “J48 algorithm” has high accuracy (correct percent) of 78.3784% which provides the banker to decide whether the loan can be given to the costumer or not. In paper [2], “loan prediction using Ensemble technique”, used “Tree model”, “Random forest”, “svm model” and combined the above three models as Ensemble model. A prototype has been discussed in paper [2] so that the banking sectors can agree/reject the loan request from their customers. The main method used is real coded genetic algorithms. The combined algorithms from the ensemble model, loan prediction can be done in an easier way. It is found that tree algorithm provides high accuracy of 81.25%. In paper [3], using R-language, an improved risk prediction clustering algorithm is used to find the bad loan customers since probability of default (PD) is the critical step for the customers who comes for a bank loan. So, a frame work for finding PD in the data set is provided by data mining technique. R- Language has the technique called as KNN (K-nearest neighbour) algorithm and it is used for performing multiple imputation calculation when there are missing values seen in the data set. The paper [4] had used tree model. It helps to find whether the banking sector people will be able to overcome the loan problem with their customers. It provides a high accuracy of 80.87%.

The paper [5] uses decision tree induction algorithm and found that the algorithm finds a best way to evaluate the credit risk. To avoid the credit risk, bankers holds the technique called as “credit score”, where it helps the lenders to keep note on who are the applicants who will able to repay the amount or probability of going into the default risks. The input given for credit evaluation was customer data, WEKA software, cibil score. The methodology used in prediction system was problem and data understanding, data filtering, system modelling and finally system evaluation. This was done on the banks existing dataset containing 1140 records and 24 attributes. At last the system was tested and helps the bankers to make a correct decision on whether to accept or reject the loan approval.

Existing Work/System:

The Bank uses predictive model technique and descriptive model technique to predict the loan approval in banks. In predictive model technique, classification and regression were used and in descriptive model technique clustering and association were used. Classifiers also implement several algorithms like naive Bayes, kNN algorithms of R language and regressors implements several algorithms like decision trees, neural networks, etc., To undergo this prediction analysis, out of all these algorithms, naive Bayes produces a most accurate classifier and the algorithms like decision tree, neural network, K-NN algorithms will be more accurate regressors. The main goal of the paper is to predict the loan classification based on the type of loan, loan applicant and the assets (property) that loan applicant holds. It was found that the decision tree algorithm gave an improved accuracy of almost 85% on doing the analysis.

Proposed Work/System:

Whenever the bank makes decision to give loan to any customers then it automatically exposes itself to several financial risks. It is necessary for the bank to be aware of the clients applying for the loan. This problem motivates to do an EDA on the given dataset and thus analysing the nature of the customer. The dataset that uses EDA undergoes the process of normalisation, missing value treatment, choosing essential columns using filtering, deriving new columns, identifying the target variables and visualising the data in the graphical format. Python is used for easy and efficient processing of data. This paper used the pandas library available in Python to process and extract information from the given dataset. The processed data is converted into appropriate graphs for better visualisation of the results and for better understanding. For obtaining the graph Matplot library is used.

Working modules and Flow chart / System Design of Proposed Work :

The architecture of the proposed model is shown in flow chart Fig.1. The major objective of this project is to derive patterns from the datasets which are used for the loan sanctioning process and create a model based on the patterns derived in the previous step. Classification data mining algorithms are used to filter out the probable loan defaulters from the list. For analysis purposes, essential inputs like gender, age, marital status, residential status, job, income, loan expectation, existing client, account balance, total debt, etc., are collected and used to find the appropriate attributes.

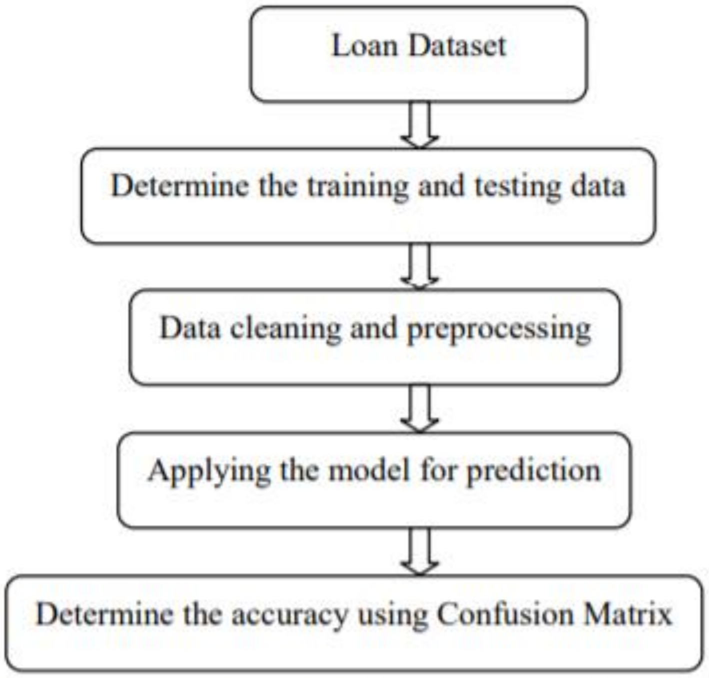

Loan Dataset
Determine the training and testing data
Data cleaning and preprocessing
Applying the model for prediction
Determine the accuracy using Confusion Matrix

Fig. 1.Architecture of the Proposed Loan Prediction Model

In the present research work, four models are implemented for loan predictions. They are, (1) Logistic Regression (LR), (2) Decision Tree (DT) (3) Support Vector Machines (SVM) and (4) Naïve Bayes (NB) method. The following section deals with the brief description of loan prediction algorithms used in the proposed method.

A. LR Model

In general, linear Regression model was used to predict the functionalities of a continuous variable, say for example ―Y‖. If the variable ―Y‖ is categorical, instead of continuous, then the LR method is adopted. The output of LR model is dichotomous i.e., binary possibilities, used for prediction of loan sanction possibilities. Properties of LR include (1) dependent variable(s) follows Bernoulli distribution and (2) Maximum likelihood is used for estimation. Further, the function f(g) is a logistic function, referred as Sigmoidal function.

B. Decision Tree

(DT) Model DT (Shown in Fig.2) is a supervised learning algorithm used to solve classification and regression problems too. Here, DT uses tree representation to solve the prediction problem, i.e., external node and leaf node in a tree represents attribute and class labels respectively. The pseudo code for DT model is depicted in the following section: Step 1: Best attribute is chosen as the tree‘s root. Step 2: Training set is divided into subsets, such that, each subset comprises similar value for an attribute. Step 3: Step 1 and Step 2 are repeated for all subsets until all the leaf nodes are traversed in a tree.

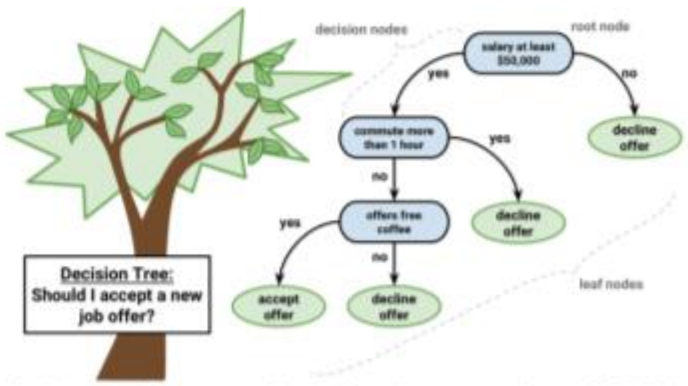


Fig. 2.shows the example in implementation of DT based approach regarding the new job offer processing the depicted form [10].

C. SVM

In this approach, each data item is plotted in a n-dimensional space, where _n' represents the number of features with each feature represented in a corresponding co-ordinates. A hyperplane is determined to distinguish the classes (possibly two) based on their features.

D. Naïve Bayes

(NB) Model The basis for NB model is Bayes Theorem (BT), where events are mutually exclusive similar to rolling a die. Moreover, the BT presumes that the input features also referred as predictors are independent in nature. Similarly, NB also presumes that the input features are independent in nature. But, this is impossible in the realistic procedures. Since this assumption leads to naïve, this algorithm is termed as Naïve Bayes algorithm. Thus, NB is a probabilistic algorithm, where the conditional probability is determined regarding the input features. On the other hand, during the dependent input features scenario, conditional probability is calculated twice resulting in improper results. Hence, for better prediction results with respect to NB model, independent input features are selected and processed [11]. The following shows the pseudo code for the proposed loan prediction method.

```
1. Load the data.
2. Determine the training and testing data.
3. Data cleaning and preprocessing.
   a) Fill the missing values with mean values regarding numerical values.
   b) Fill the missing values with mode values regarding categorical variables.
   c) Outlier treatment.
4. Apply the modeling for prediction.
```

a) Removing the load identifier.

b) Create the target variable (based on the requirement). In this approach, target variable is loan-status.

c) Create a dummy variable for categorical variable (if required) and split the training and testing data for validation.

d) Apply the model • LR method • DT method • RF method • SVM method

5. Determine the accuracy followed by confusion matrix.

Description of each modules with appropriate diagram and explanation:

A. Annual Income Vs Purpose Of Loan

In this Figure 1, the X axis represents the purpose of loan i.e. the purpose for which the loan is applied. Debt consolidation, home improvement is some of the purposes. High, moderate and low represents the annual income of people who fall in the range as below. Low represents the annual income of people between the range of minimum to 10 lakhs and Moderate represents the annual income of people between 10 lakhs and 25 lakhs and High represents the annual income of people above 25 lakhs. By these criteria, a new column called Category is derived.

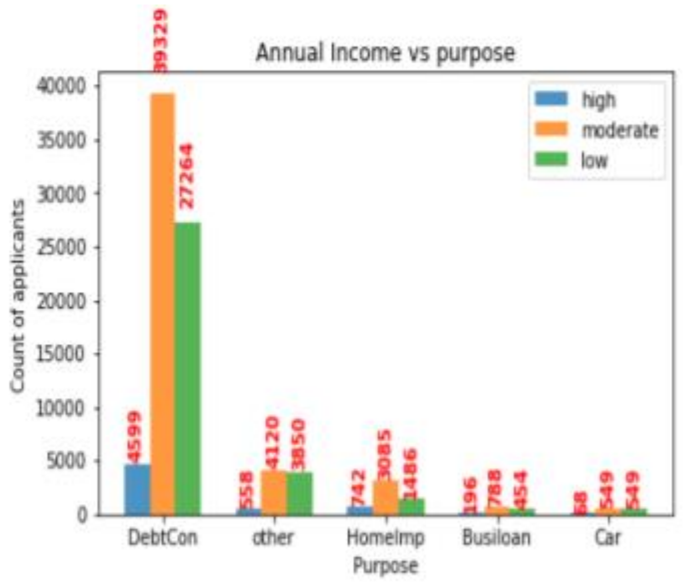


Fig 1. Annual Income vs purpose

Thus, grouping the Category that is high, moderate and low.

Inference from the Figure 1 is as follows:

• People in moderate category seek loan in the higher numbers.

• The field debt consolidation shows the highest distribution.

• Low and moderately categorized applicants try for other purpose and car loans equally.

B. Trust Customer Classification

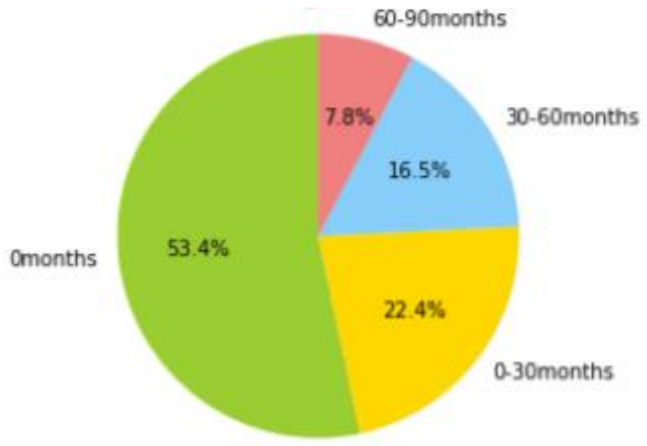


Fig 2. Trust Customers

From the Figure 2 it is inferred as follows:

• There are many customers who does not have delinquents, has applied for the loan which intimates or indirectly conveys that the applicant has some chances to get approval of the loan as the applicant have no delinquents. The result is about 53.3% of applicants.

• And it is also inferred that the number of people applied for loan gradually decreases with the increase in the delinquent months. This shows that, the applicant has minimum chances to get the approval of loans

C. Loan Term Vs Delinquent Months

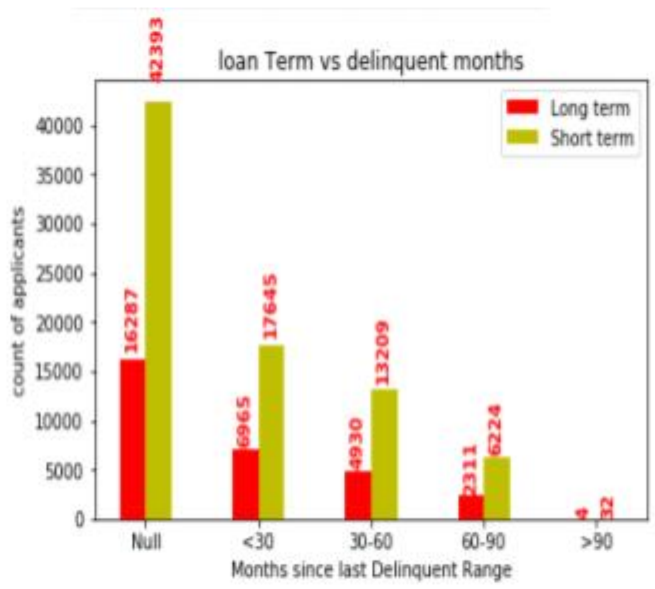


Fig 3. loan term vs delinquent months

- Figure 3 deals with the customers who can pay the loan within term period against customers who cannot repay their monthly depts within the particular term.

From the Figure 3 it can be concluded that:

• This analysis can find a higher number of customers who are able to repay without deliquiates and for short term.

• Almost all applicants who are even delinquent more than 90 months prefers only short term.

• Applicants who delinquent more than 90 months are less in number and it indirectly conveys that their loan will never be sanctioned and if its yes, the applicant will not be able to pay it back.

D. Loan Term Vs Credit Category

From the categories poor, fair, good, very good and undefined, the value the credit category is found, as applicants without credit score fall into the category undefined, people have credit score between 300-850 between these there are some categories such as credit score between 300 and 579 falls in poor, credit score between 580 and 669 falls in category fair and the credit score between 670 and 739 is good and above this is considered to be very good.

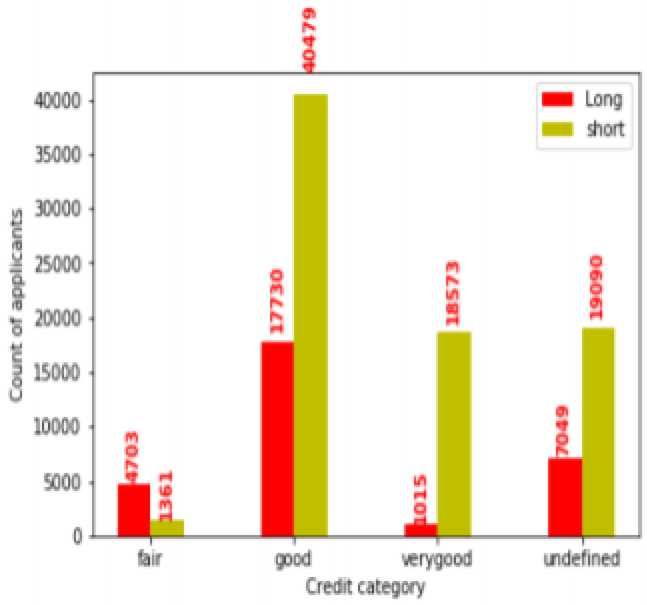


Fig 4. Loan term vs Credit category

Figure 4 spectacles the repayment period of the loan versus credit score under various categories by grouping the derived column credit category and loan term. Figure 4 have deduced the following:

- Customers with good and very good credit score prefer for short term payback period in contradiction to customers with fair credit score.

- People applying loan for first time prefer short term loans because lender doesn't do a credit check so that the applicant avail loans easily.

E. Loan Term Vs Years In Current Job

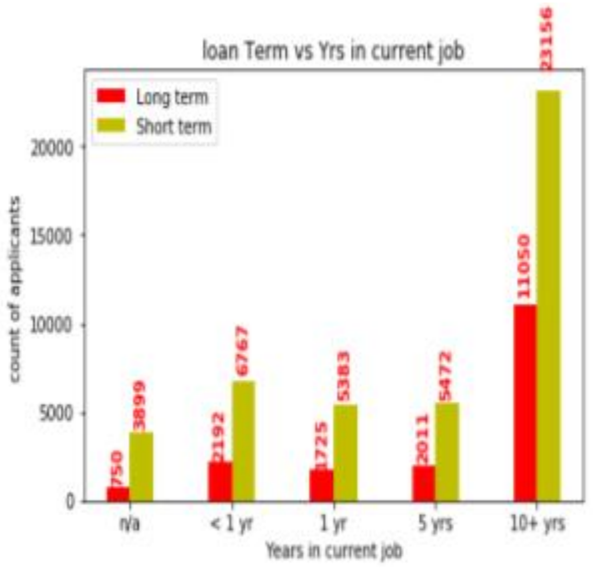


Fig 5. Loan term vs Years in current job

Figure 5 displays the count of applicants who have various years of experience in current job against the term period of repayment of loans. From the Figure 5 it is concluded that:

- Applicants who have various years of experience in the same job claim the loan for a short period of time.
- Also, the applicants who are freshers claim the loan for a short period of time.
- This also infers that, long term loans are borrowed by people who are yet to start their own business and can repay only after this business set well and brings profit.
- Long term goals carry a greater risk to the money lenders and thus not very easily approved by the banks.

F. Loan Payment Chances Vs Home Ownership

Loan payment chances which have been classified into canpay, maypay and not payable. From Current credit balance, subtracting the monthly debt of that person, one can find whether the person would be able to repay the loan or not. By subtracting, if applicants have less than 50,000 balance, the applicant is considered in the not payable category and applicants having balance between 50,000 and 3 lakhs are considered to be may pay category and above than that the applicant are considered to be can pay category.

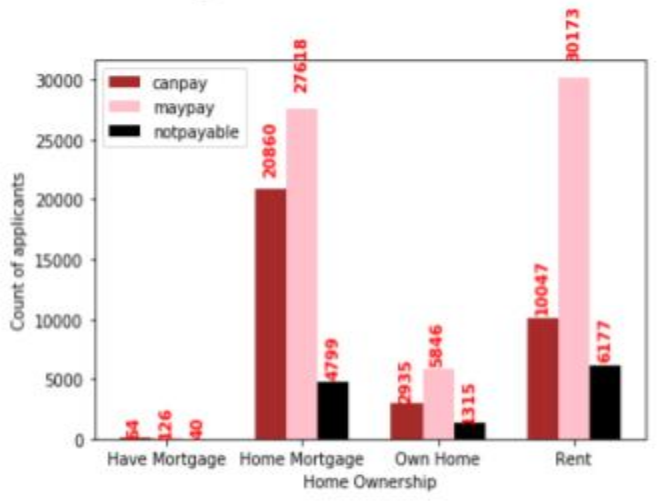


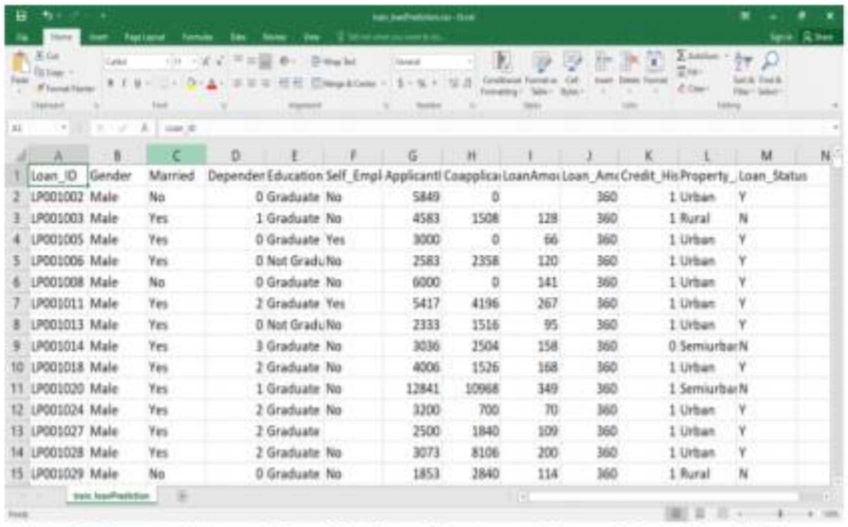
Fig 6. Loan payment chances vs Home ownership

From the Figure 6 it is concluded that,

- People living in rent home fall under not payable category among ownership credentials
- Applicants who have kept home in mortgage apply loan in highest numbers.

Result and discussion:

At the start, the dataset was cleaned. Then exploratory data analysis and feature engineering were performed. Then a model was created which predicted whether the applicant would repay the loan or not. Whenever the bank makes decision to give loan to any customers then it automatically exposes itself to several financial risks. It is necessary for the bank to be aware of the clients applying for the loan. This problem motivates to do an EDA on the given dataset and thus analyzing the nature of the customer. The dataset that uses EDA undergoes the process of normalization, missing value treatment, choosing essential columns using filtering, deriving new columns, identifying the target variables and visualizing the data in the graphical format. Python is used for easy and efficient processing of data. This paper used the pandas library available in Python to process and extract information from the given dataset. The processed data is converted into appropriate graphs for better visualization of the results and for better understanding. For obtaining the graph Matplot library is used.



Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantI	Coapplica	LoanAmou	Loan_Am	Credit_His	Property_	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
LP001014	Male	Yes	3	Graduate	No	3036	2504	158	360	0	Semiurbar	N
LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurbar	N
LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N

Fig. 3.Input Loan Prediction Dataset from Kaggle [12]

Figure 3 shows the details of dataset collected from Kaggle source [12]. The feature in the dataset includes,

1. Loan_Id
2. Gender
3. Marital Status
4. Number of dependents
5. Educational Profile
6. Employment Status
7. Applicant‘s Income
8. Co-Applicant‘s Income
9. Loan Amount

10. Loan Tenure

11. Credit History

12. Property Area

13. Loan Status

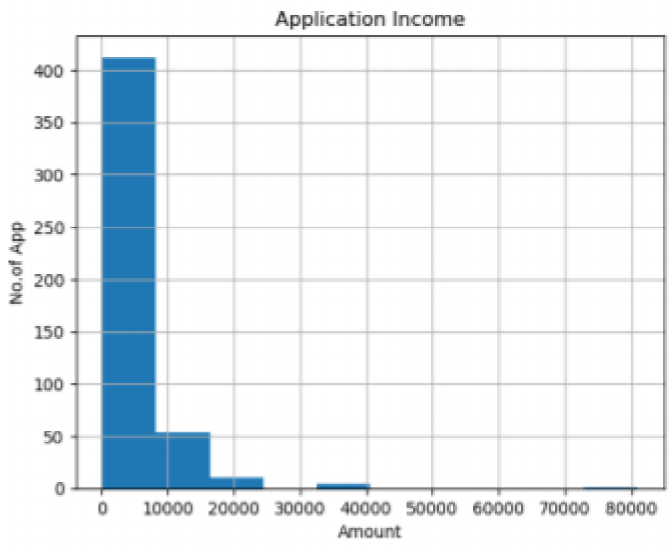


Fig. 4.a. Application Income

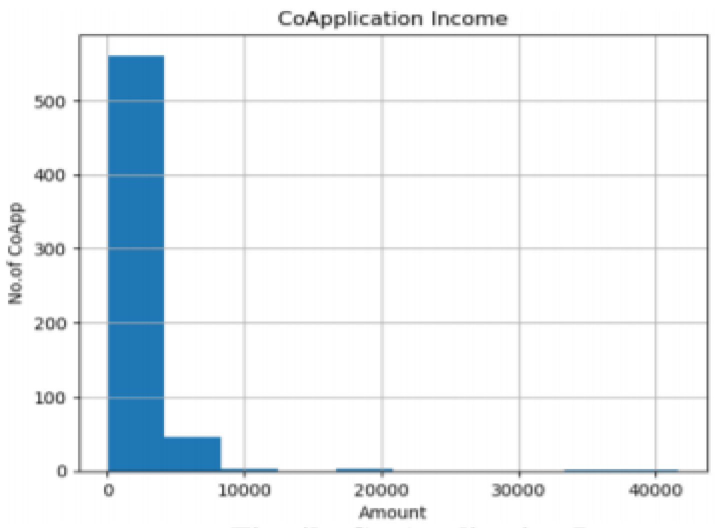


Fig. 4b. Co-Application Income

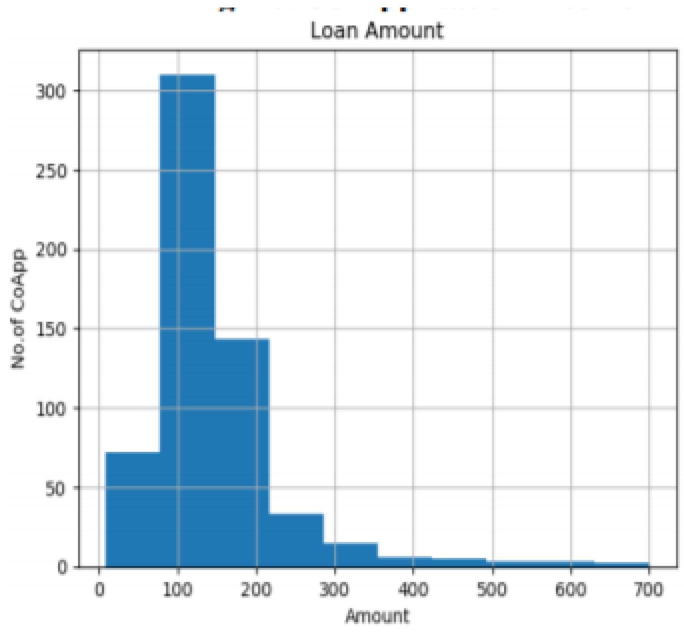


Fig 4c. Loan Amount

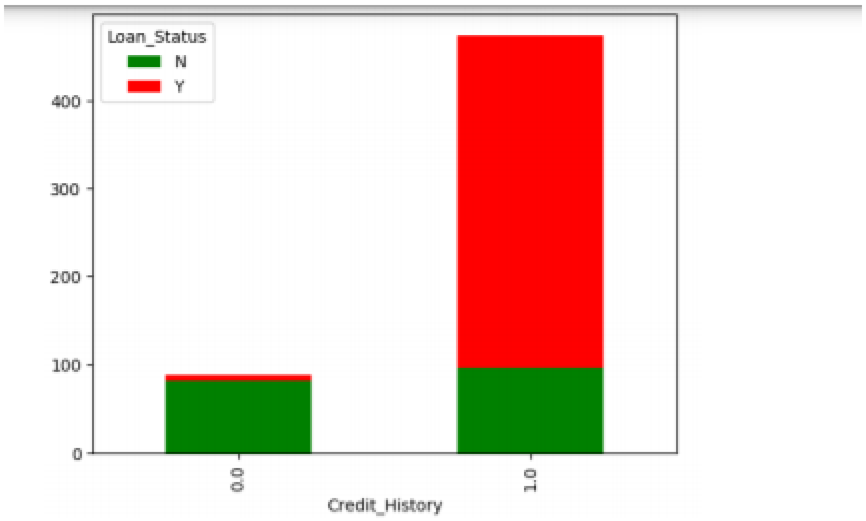


Fig.4d Credit History

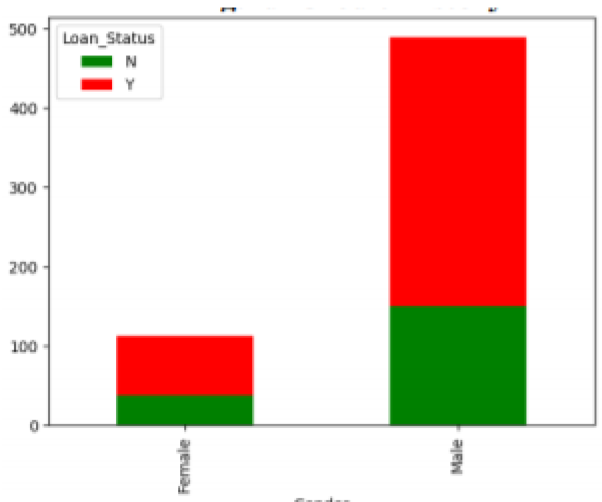


Fig. 4e. Gender Loan Status

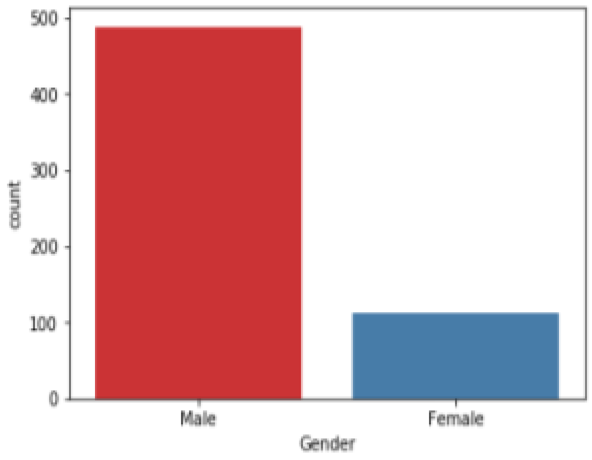


Fig.4f Graph for Gender

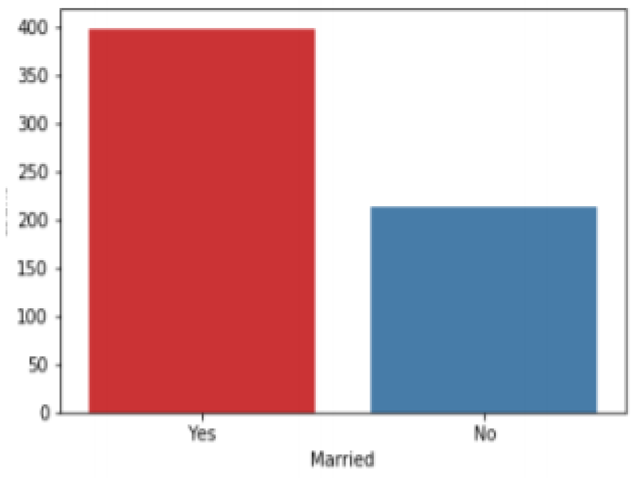


Fig .4g Graph for Relation Status

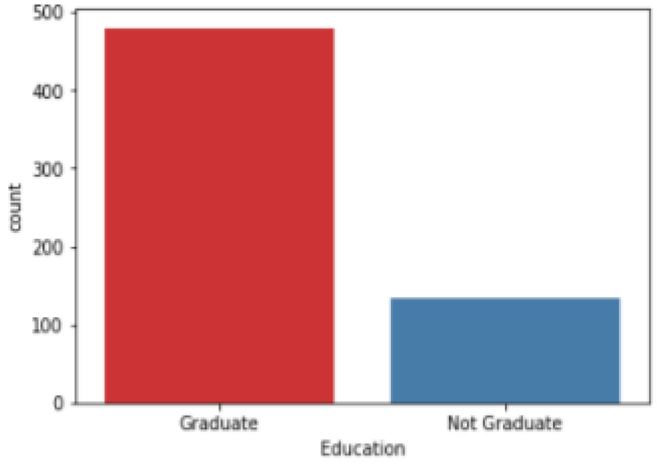


Fig.4h Graph for Graduate and not graduate

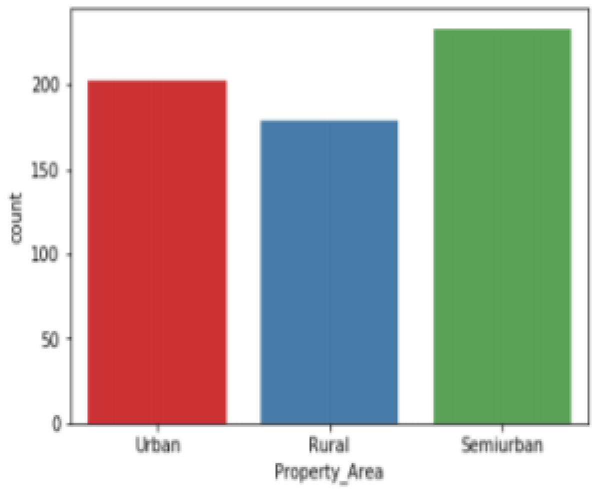


Fig.4i. Graph for Rural or Urban or Semi urban

As represented in the pseudo code of the proposed loan prediction model, after loading the data from the dataset, the training and testing data are determined followed by the data cleaning and preprocessing procedures. Later the prediction models are applied and the performance is determined using the Confusion Matrix (CM) method. The Confusion Matrix (CM) is used to analyze and determine the performance of the proposed loan prediction model (Shown in Table I and II). Figure 5 shows the CM parameters summarized from [13-14]. The interpretation in the CM is as follows:

• True Positive (TP), when both the actual and predicted values are positive (1)

- True Negative (TN), when both the actual and predicted values are negative (0)

- False Positive (FP), when the actual value is negative and the predicted value is positive (1)

- False Negative (FN), when the actual value is positive (1) and the predicted value is negative (0)

Table I Predicted Value

		Predicted Values	
		Negative (0)	***Positive (1)***
Actual Values	Negative (0)	TN	FP
	Positive (1)	FN	TP

Table II Performance Comparison

Parameter	**Loan Prediction Models**			
	LR	***DT***	***SVM***	***NB***
Loan Prediction Accuracy	78.91	71.92	65.27	80.42

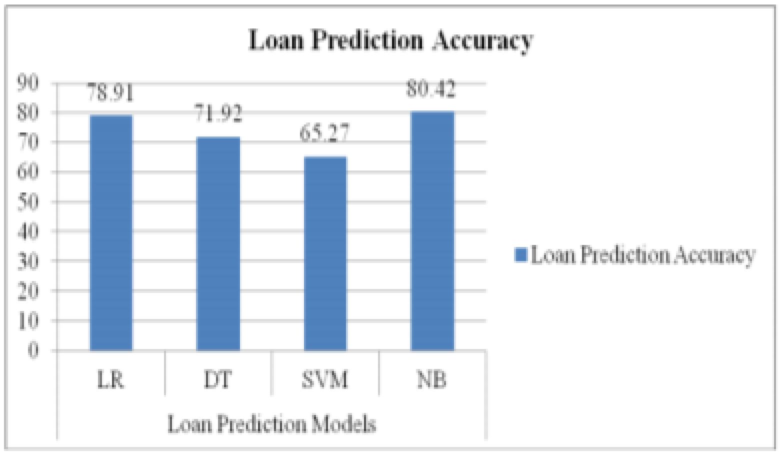


Fig.5. Comparison of Prediction Accuracy

Complete Program/code:

Links:

Jupyter Notebook:

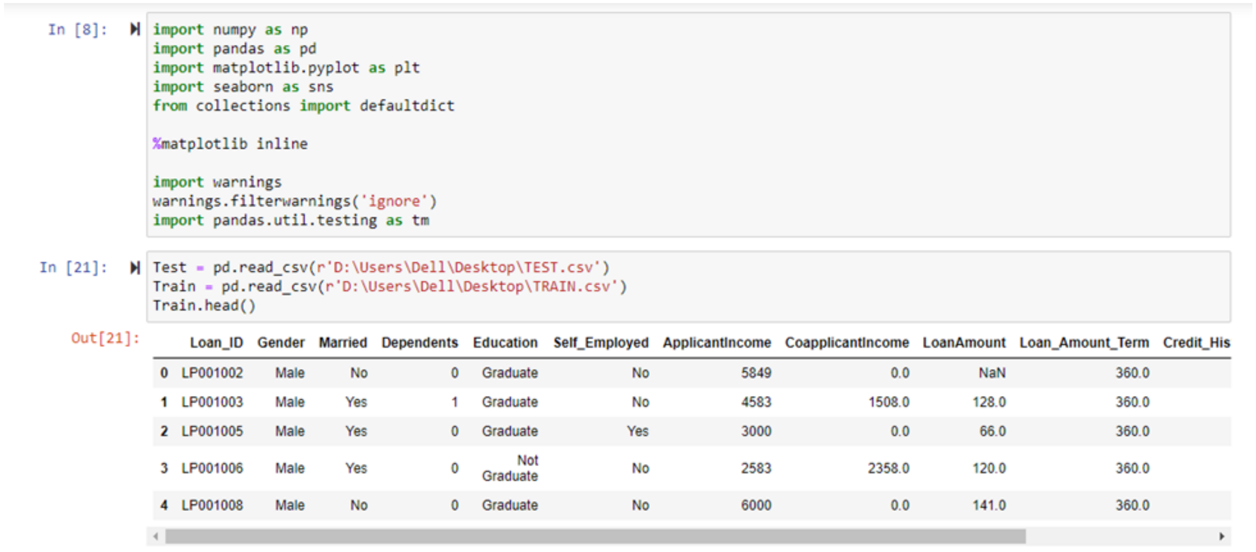
http://localhost:8888/notebooks/3040%20PROJECT.ipynb#

Tableau:

https://public.tableau.com/profile/keerthana.madhavan.19mia1073#!/vizhome/3040LOANPREDICTIONTRAINTABLEAU/Sheet19

https://public.tableau.com/profile/keerthana.madhavan.19mia1073#!/vizhome/3040LOANPREDICTIONTESTTABLEAU/Sheet3

Screenshots of the code and the output:

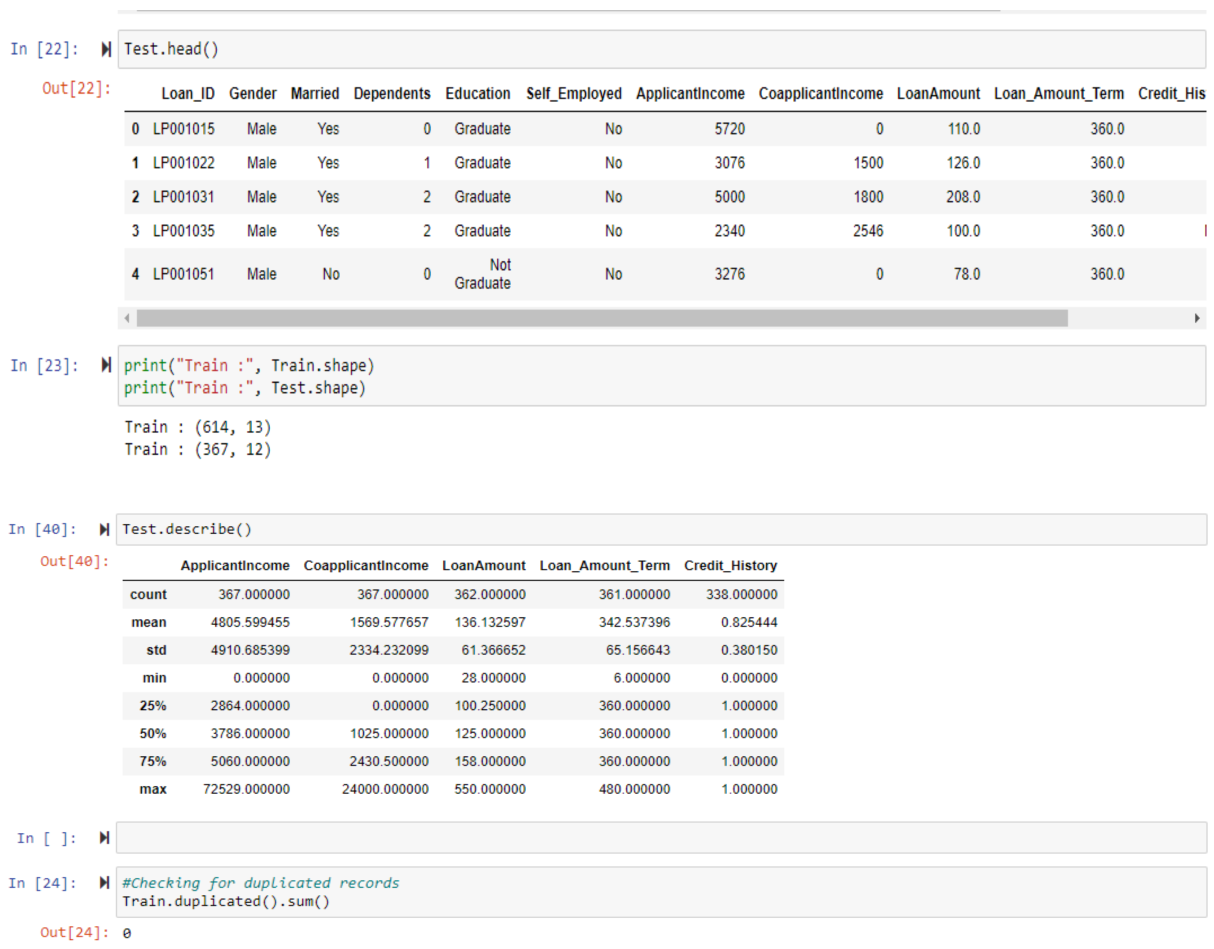


```
In [8]:  import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from collections import defaultdict

         %matplotlib inline

         import warnings
         warnings.filterwarnings('ignore')
         import pandas.util.testing as tm
```

```
In [21]: Test = pd.read_csv(r'D:\Users\Dell\Desktop\TEST.csv')
         Train = pd.read_csv(r'D:\Users\Dell\Desktop\TRAIN.csv')
         Train.head()
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_His
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	



```
In [22]: Test.head()
```

Out[22]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_His
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	

```
In [23]: print("Train :", Train.shape)
         print("Train :", Test.shape)

         Train : (614, 13)
         Train : (367, 12)
```

```
In [40]: Test.describe()
```

Out[40]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	367.000000	367.000000	362.000000	361.000000	338.000000
mean	4805.599455	1569.577657	136.132597	342.537396	0.825444
std	4910.685399	2334.232099	61.366652	65.156643	0.380150
min	0.000000	0.000000	28.000000	6.000000	0.000000
25%	2864.000000	0.000000	100.250000	360.000000	1.000000
50%	3786.000000	1025.000000	125.000000	360.000000	1.000000
75%	5060.000000	2430.500000	158.000000	360.000000	1.000000
max	72529.000000	24000.000000	550.000000	480.000000	1.000000

```
In [ ]:

In [24]: #Checking for duplicated records
         Train.duplicated().sum()

Out[24]: 0
```

So we observed that there are 614 records in the Training Dataset and 367 records in the Testing Dataset.

Training Dataset has Loan_ID, Gender, Married, Dependents, Education, Self_Employed, Property_Area and Loan_status as object types. Object type in pandas is similar to strings. ApplicantIncome field is of integer type. The other three fields namely CoapplicantIncome, Loan_Amount_Term and Credit_History are floating point type.

Testing Dataset is also very similar to Training, except the fact that CoapplicantIncome is of Integer Type. And there is no column as Loan_status since thats what we have to predict by creating a model.

Univariate Analysis: Univariate Analysis involves analysis of one variable at a time. Let's say "Gender" then we will analysis only the "Gender" field in the dataset. The analysis is usually summarised in the form of count. For visualisation, we have many options such as frequency tables, bar graphs, pie charts, histograms etc.

Categorical Varaibles : Categorical variables are those data fields that can be divided into groups. In this case, Gender(Male OR Female), Married(Yes Or No), Education(Graduate Or Not Graduate), Property_Area (Urban, Semiurban Or Rural), Self_Employed(Yes Or No), Loan_Status(Y Or N) are the categorical variables.

```
In [42]:  pd.crosstab(Train['Credit_History'], Train['Loan_Status'], margins=True)
```

Out[42]:

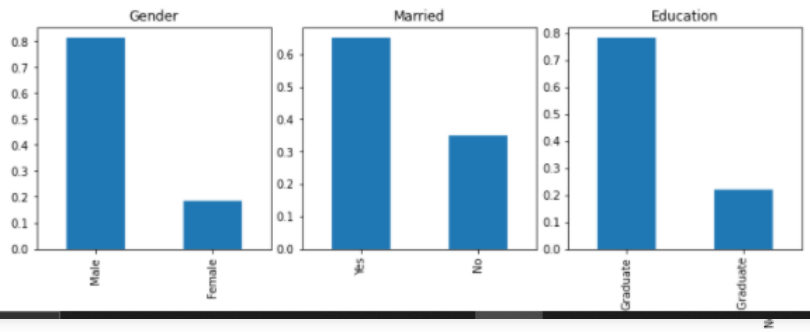
Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

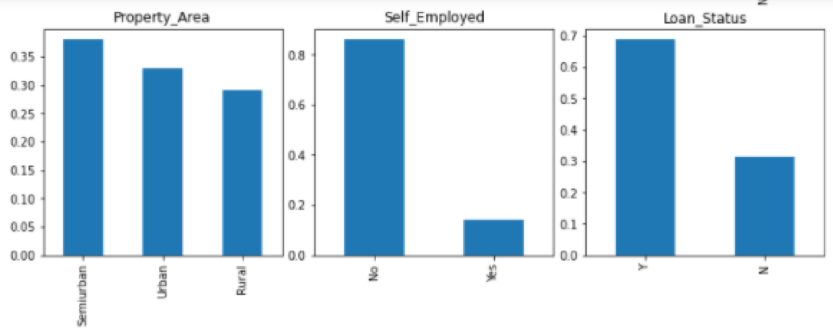
```
In [43]:  pd.crosstab(Test['Credit_History'], Train['Loan_Status'], margins=True)
```

Out[43]:

Loan_Status	N	Y	All
Credit_History			
0.0	23	36	59
1.0	78	201	279
All	101	237	338

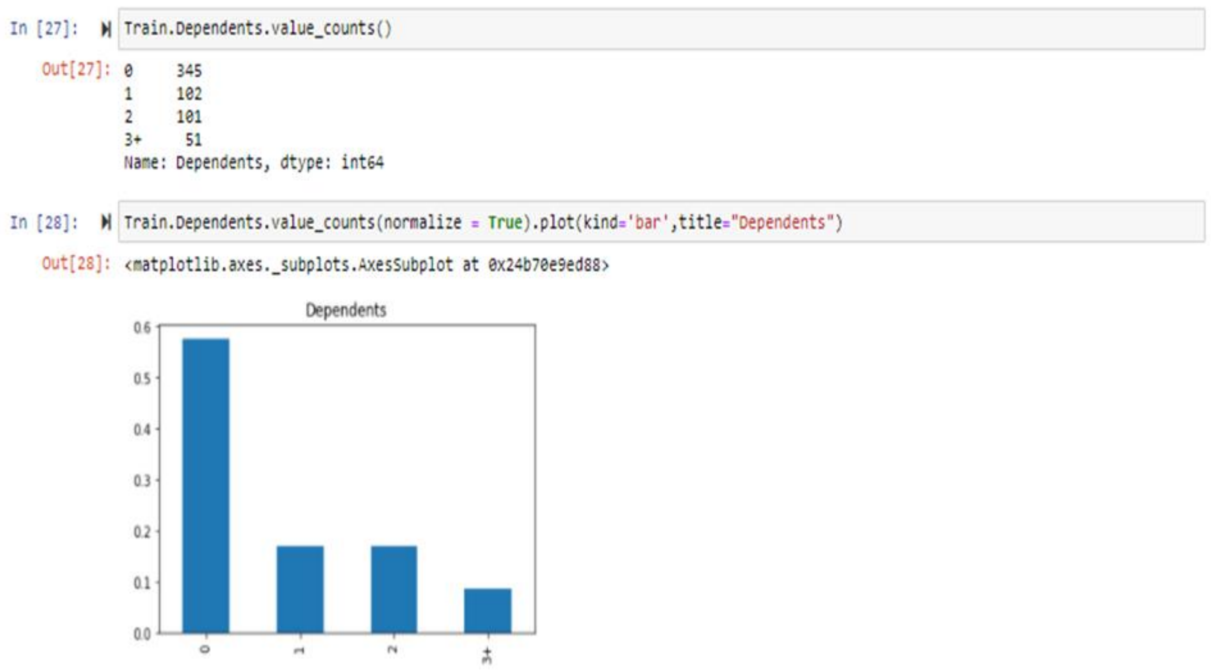
```
In [25]:  plt.subplot(231)
          Train.Gender.value_counts(normalize=True).plot(kind = 'bar', title = "Gender",figsize=(10,8))
          plt.tight_layout(pad=0.5)

          plt.subplot(232)
          Train.Married.value_counts(normalize=True).plot(kind = 'bar', title = "Married")
          plt.tight_layout(pad=0.5)

          plt.subplot(233)
          Train.Education.value_counts(normalize=True).plot(kind = 'bar', title = "Education")
          plt.tight_layout(pad=0.5)

          plt.subplot(234)
          Train.Property_Area.value_counts(normalize=True).plot(kind = 'bar',title = "Property_Area")
          plt.tight_layout(pad=0.5)

          plt.subplot(235)
          Train.Self_Employed.value_counts(normalize=True).plot(kind = 'bar',title = "Self_Employed")
          plt.tight_layout(pad=0.5)

          plt.subplot(236)
          Train.Loan_Status.value_counts(normalize=True).plot(kind = 'bar',title = "Loan_Status")
          plt.tight_layout(pad=0.5)
```





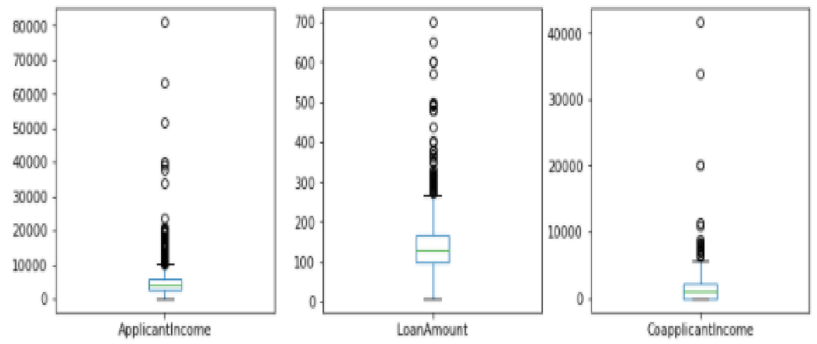
Findings from Univariate analysis of categorical variables :

- 80% of loan applicants are from male in the training dataset.
- Nearly 70% are married
- About 75% of loan applicants are graduates
- Highest number of applicants are from Semiurban areas, followed by urban areas.
- Nearly 85-90% loan applicants are self employed
- Loan has been approved for more than 65% of applicants.

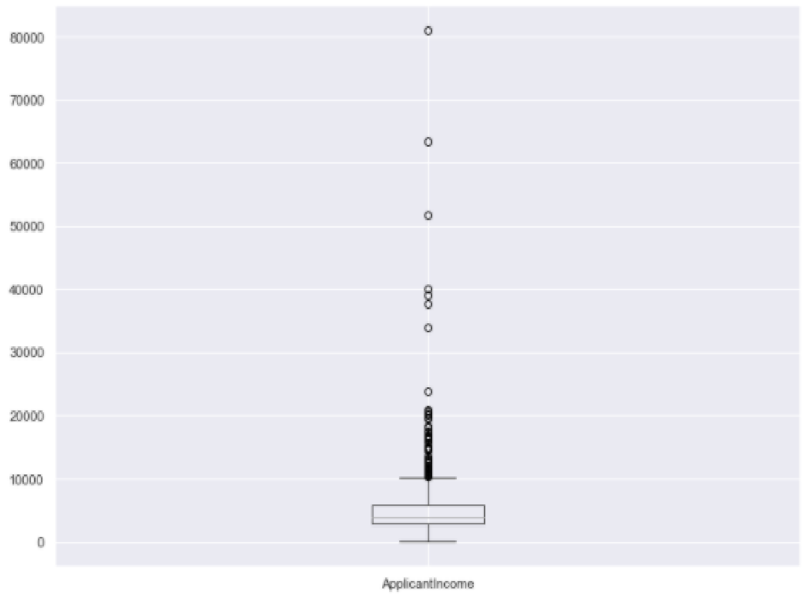


```
In [27]:  Train.Dependents.value_counts()

Out[27]: 0     345
         1     102
         2     101
         3+     51
         Name: Dependents, dtype: int64

In [28]:  Train.Dependents.value_counts(normalize = True).plot(kind='bar',title="Dependents")

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x24b70e9ed88>
```

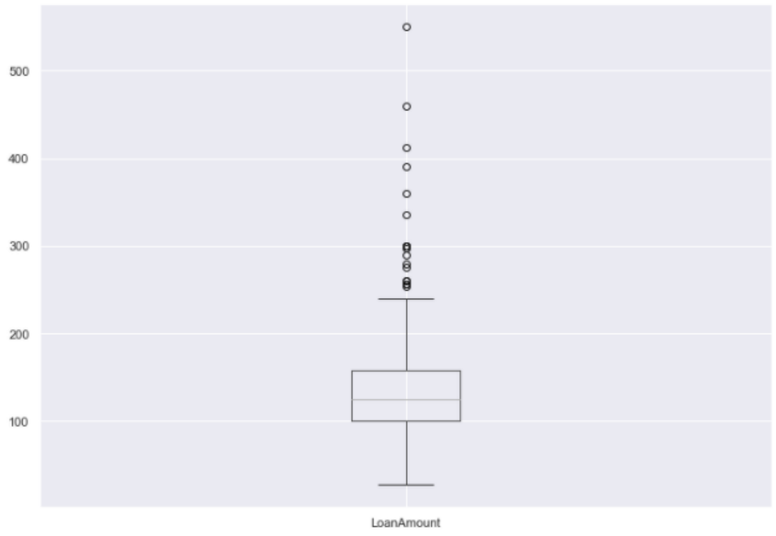
```
In [29]:  #bins = np.linspace(Train.ApplicantIncome.min(),Train.ApplicantIncome.max(),10)
          plt.subplot(231)
          Train['ApplicantIncome'].plot.box(figsize=(10,6))
          plt.tight_layout(pad=0.5)

          plt.subplot(232)
          Train['LoanAmount'].plot.box()
          plt.tight_layout(pad=0.5)

          plt.subplot(233)
          Train['CoapplicantIncome'].plot.box()
          plt.tight_layout(pad=0.5)
```



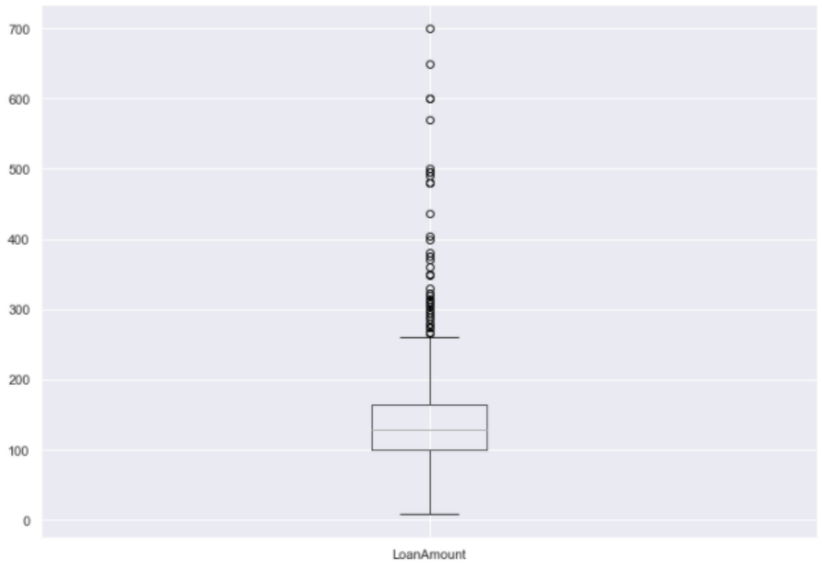
```
In [44]:  Train.boxplot(column='ApplicantIncome')

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x24b6f9a2d48>
```



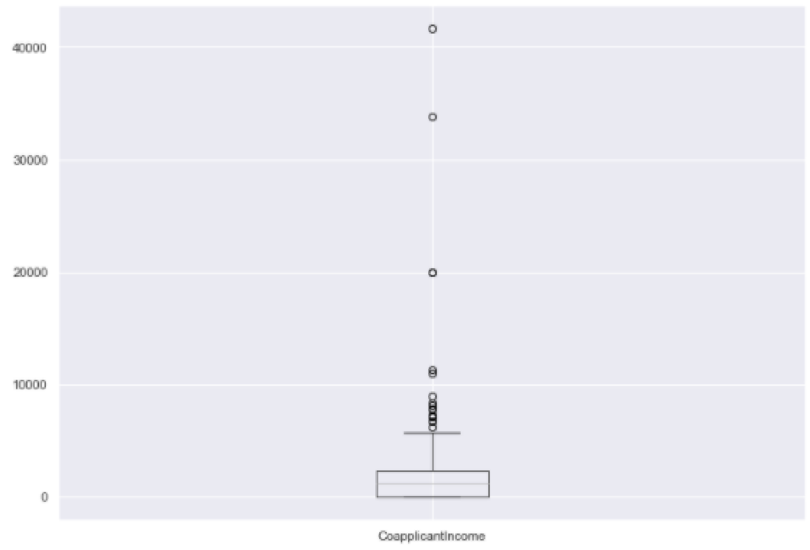
```
In [135]:  Test.boxplot(column='LoanAmount')
Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x24b7317f208>
```



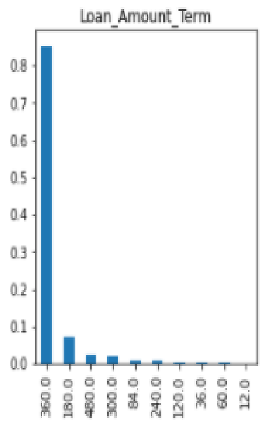
```
In [136]:  Train.boxplot(column='LoanAmount')
Out[136]: <matplotlib.axes._subplots.AxesSubplot at 0x24b731db8c8>
```



```
In [46]:  Train.boxplot(column='CoapplicantIncome')
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x24b6ff48188>
```



```
In [30]:  Train.Credit_History.value_counts(normalize = 'True').plot(kind = 'bar', title='Credit_History')
          plt.tight_layout(pad=0.5)

          plt.subplot(122)
          Train.Loan_Amount_Term.value_counts(normalize = 'True').plot(kind = 'bar', title='Loan_Amount_Term')
          plt.tight_layout(pad=0.5)
```



Conclusions from univariate analysis of continous variables.

- 80% of applicants have credit history of 1
- More than 80% of loans are taken for 360 days or above.

- The applicantIncome is mostly between 10000-40000 with some outliers.
- CoapplicantIncome is lesser than applicantIncome and is within the 5000-15000, with some outliers.
- Loan Amount is mostly concentrated between 300-500.
- We might have to remove outliers from applicantIncome and CoapplicantIncome. But for that we need to find reasons for outliers.

Bivariate Analysis : Bivariate analysis is basically finding some kind of emperical relationship between two variables. Let's say ApplicantIncome and Loan_Status. Before performing any kind of analysis, we must create an hypothesis.This hypothesis will act as a guiding light, where to look and analyse.

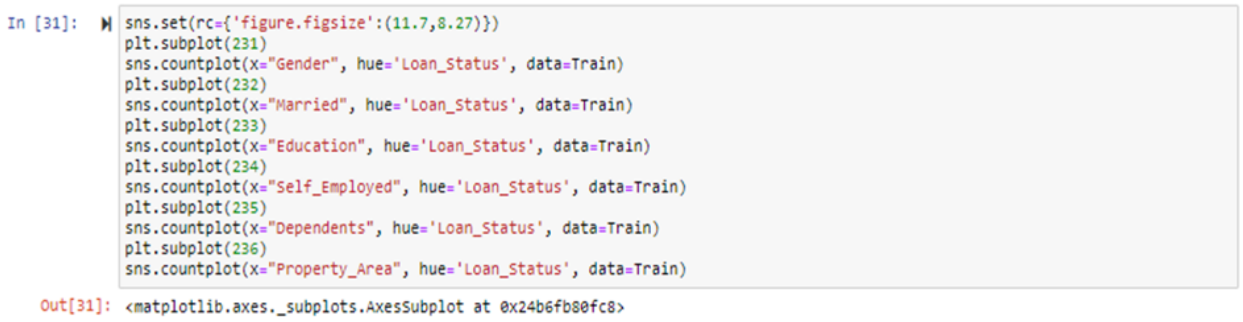
For the purpose of this analysis,

We have the following **hypothesis** in my mind.

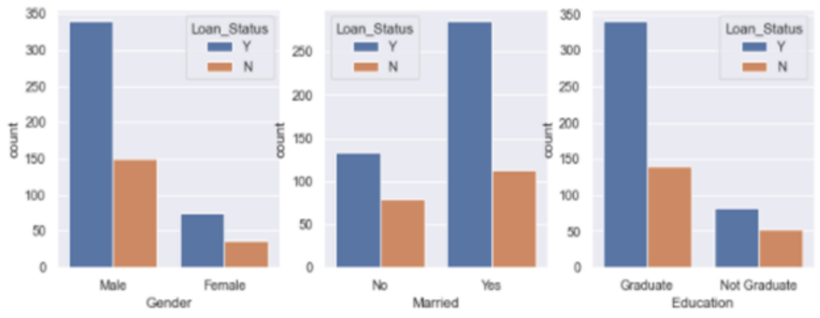
- Applicants with higher income might have more chances of getting their loans approved.
- Applicants with less number of dependents higher coapplicantIncome might have more chances of getting loan approvals.
- Applicants who are graduates, tend to earn more and hence have higher loan approval rates.
- Applicants who are married, might seem more responsible hence higher loan approval chances.

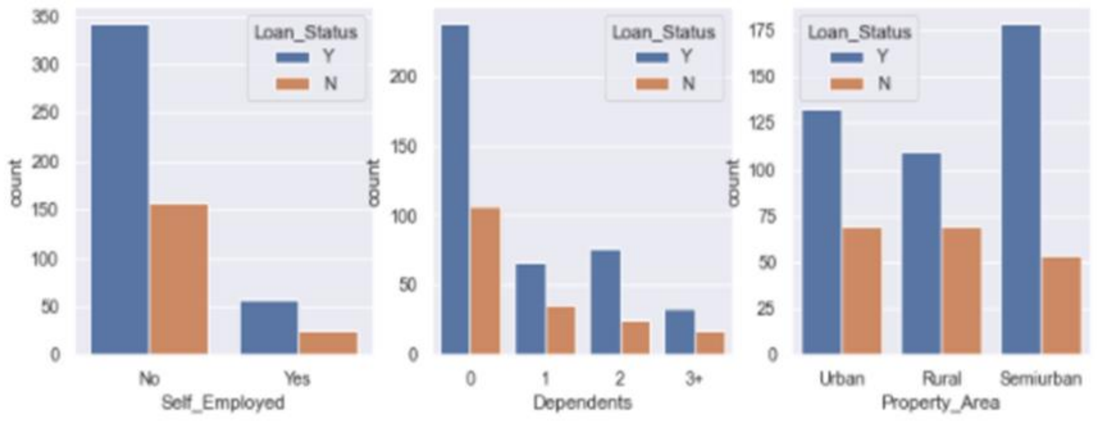
- Applicants who are not self-employed, might have a higher chances of loan approval as they tend to have constant source of income. There is less uncertainity.
- Candidates with property in urban areas might have higher chances of loan approval, since the cost of collateral would be high.
- Good credit history should definitely correlate with loan approval.
- For Gender let's say women tend to be more responsible and hence high approval rates.

Now, let's check if this hypothesis, is correct or not for this dataset.



```
In [31]:  sns.set(rc={'figure.figsize':(11.7,8.27)})
           plt.subplot(231)
           sns.countplot(x="Gender", hue='Loan_Status', data=Train)
           plt.subplot(232)
           sns.countplot(x="Married", hue='Loan_Status', data=Train)
           plt.subplot(233)
           sns.countplot(x="Education", hue='Loan_Status', data=Train)
           plt.subplot(234)
           sns.countplot(x="Self_Employed", hue='Loan_Status', data=Train)
           plt.subplot(235)
           sns.countplot(x="Dependents", hue='Loan_Status', data=Train)
           plt.subplot(236)
           sns.countplot(x="Property_Area", hue='Loan_Status', data=Train)

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x24b6fb80fc8>
```





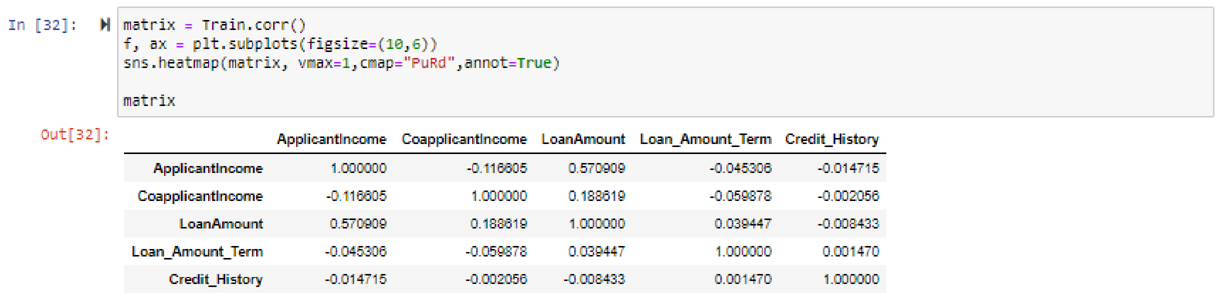
Insights :

- There is not a substantial difference between male and female approval rates.
- Married applicants have a slightly higher chances of loan approval.
- Graduates have higher chance of loan approval compared to non-graduates.
- Self_Employed employees have slightly lower chances of loan approval but the situation is not that bad.
- Applicants with no dependents or 2 dependents have higher chances of approval. But this does not correlate well.
- Applicants with properties in semi-urban areas have higher loan approval rates.

Multivariate Analysis : A multivariate analysis involves finding relationship between more than two variables.

Hypothesis :

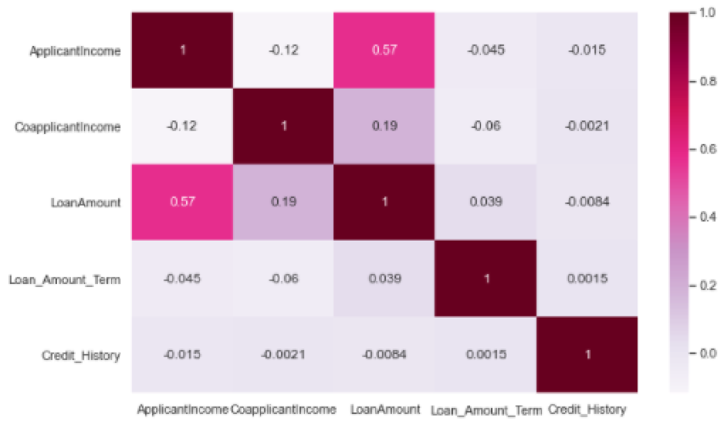
- An applicant who is a graduate, will have more income and hence might have higher changes of getting loan approval.
- Likewise we will try to figure out relationship between the variables in pair of three or more.



```
In [32]:  matrix = Train.corr()
          f, ax = plt.subplots(figsize=(10,6))
          sns.heatmap(matrix, vmax=1,cmap="PuRd",annot=True)

          matrix
```

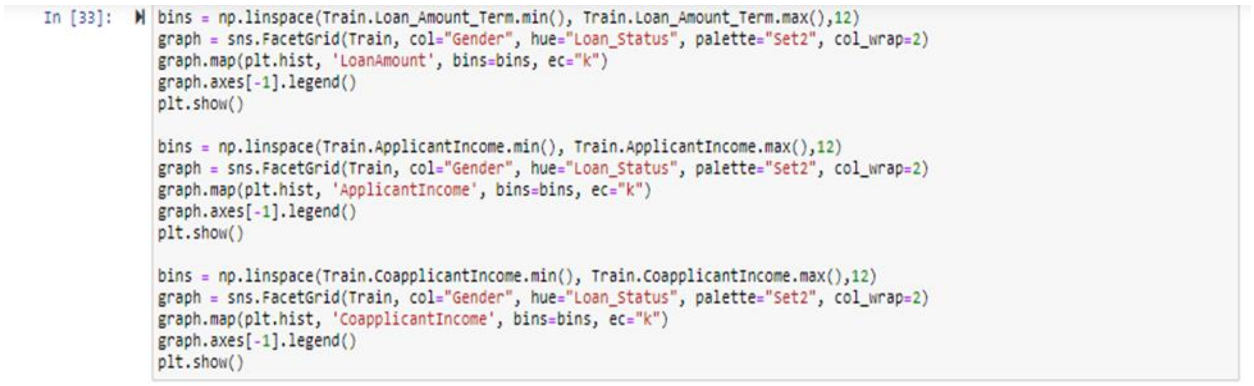
Out[32]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000

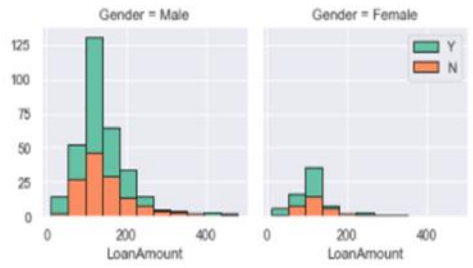


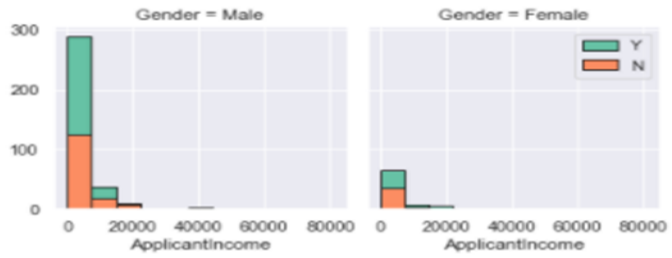
Now lets look at the correlation between all the numerical variables. We will use the heat map to visualize the correlation. Heatmaps visualize data through variations in coloring. The variables with darker color means their correlation is more.

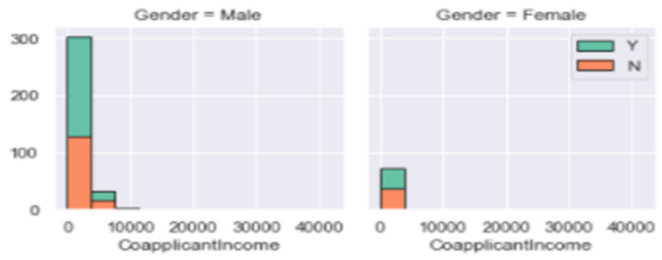
We see that the most correlated variables are (ApplicantIncome - LoanAmount) and (Credit_History - Loan_Status).



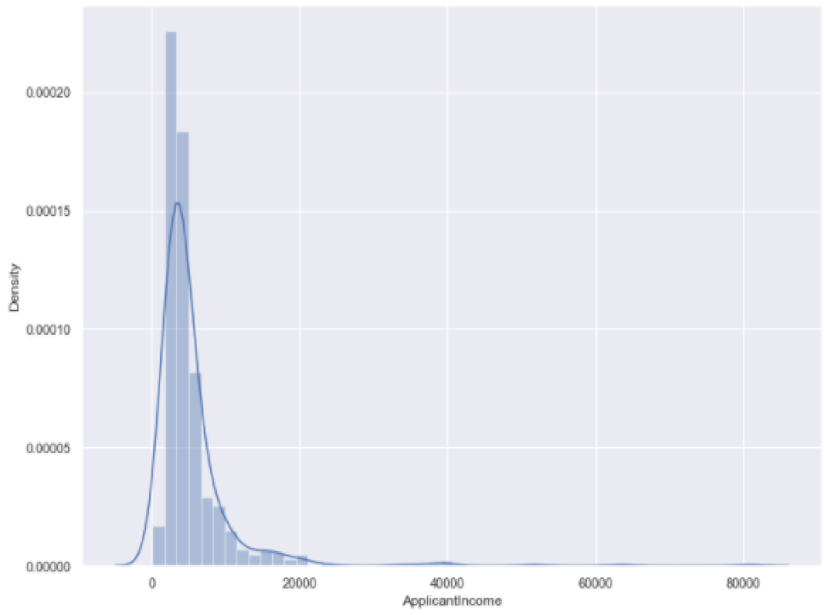
```
In [33]:  bins = np.linspace(Train.Loan_Amount_Term.min(), Train.Loan_Amount_Term.max(),12)
          graph = sns.FacetGrid(Train, col="Gender", hue="Loan_Status", palette="Set2", col_wrap=2)
          graph.map(plt.hist, 'LoanAmount', bins=bins, ec="k")
          graph.axes[-1].legend()
          plt.show()

          bins = np.linspace(Train.ApplicantIncome.min(), Train.ApplicantIncome.max(),12)
          graph = sns.FacetGrid(Train, col="Gender", hue="Loan_Status", palette="Set2", col_wrap=2)
          graph.map(plt.hist, 'ApplicantIncome', bins=bins, ec="k")
          graph.axes[-1].legend()
          plt.show()

          bins = np.linspace(Train.CoapplicantIncome.min(), Train.CoapplicantIncome.max(),12)
          graph = sns.FacetGrid(Train, col="Gender", hue="Loan_Status", palette="Set2", col_wrap=2)
          graph.map(plt.hist, 'CoapplicantIncome', bins=bins, ec="k")
          graph.axes[-1].legend()
          plt.show()
```





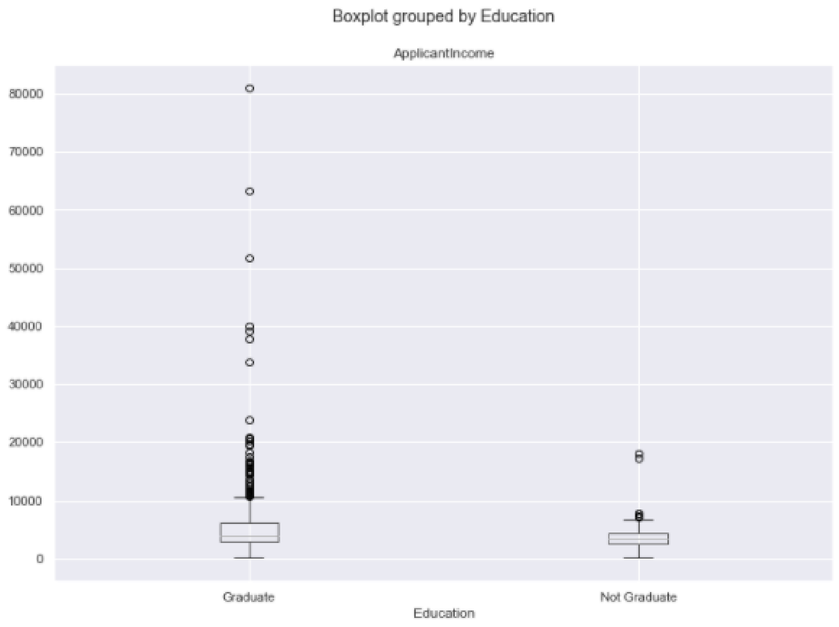


```
In [47]:  sns.distplot(Train['ApplicantIncome'])
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x24b701d6d08>
```



This is the distribution of applicant income

```
In [48]:  Train.boxplot(column='ApplicantIncome', by='Education')
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x24b6f9ff1c8>
```



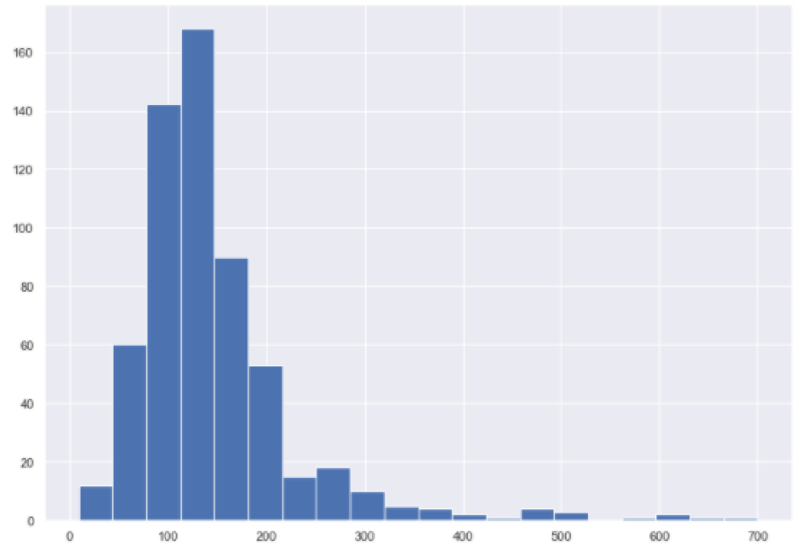
The distributions are quite similar but we can see that the graduates have more outliers which means that the people with huge income are most likely well educated.

Bivariate Analysis

These are the hypotheses that we generated earlier:

- Applicants with high incomes should have more chances of loan approval.
- Applicants who have repaid their previous debts should have higher chances of loan approval.
- Loan approval should also depend on the loan amount. If the loan amount is less, the chances of loan approval should be high.
- Lesser the amount to be paid monthly to repay the loan, the higher the chances of loan approval.

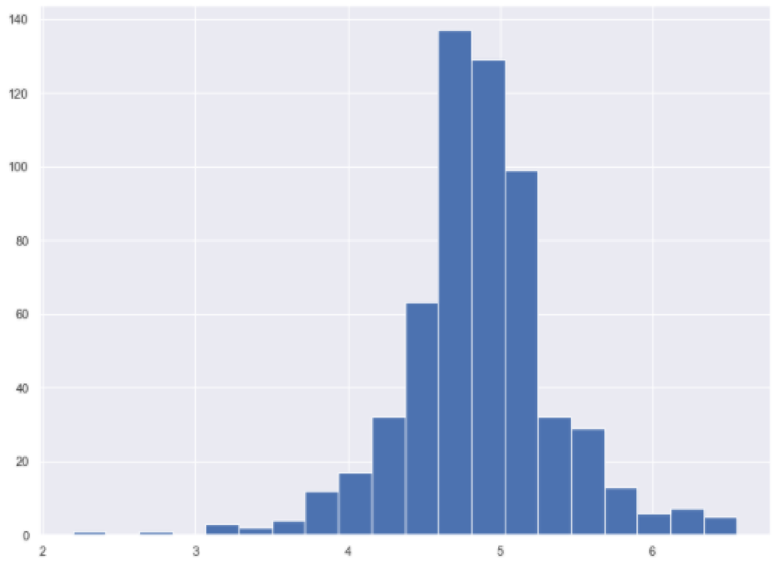
```
In [49]:  Train['LoanAmount'].hist(bins=20)

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x24b700b9488>
```



This a histogram for a loanAmount variable.

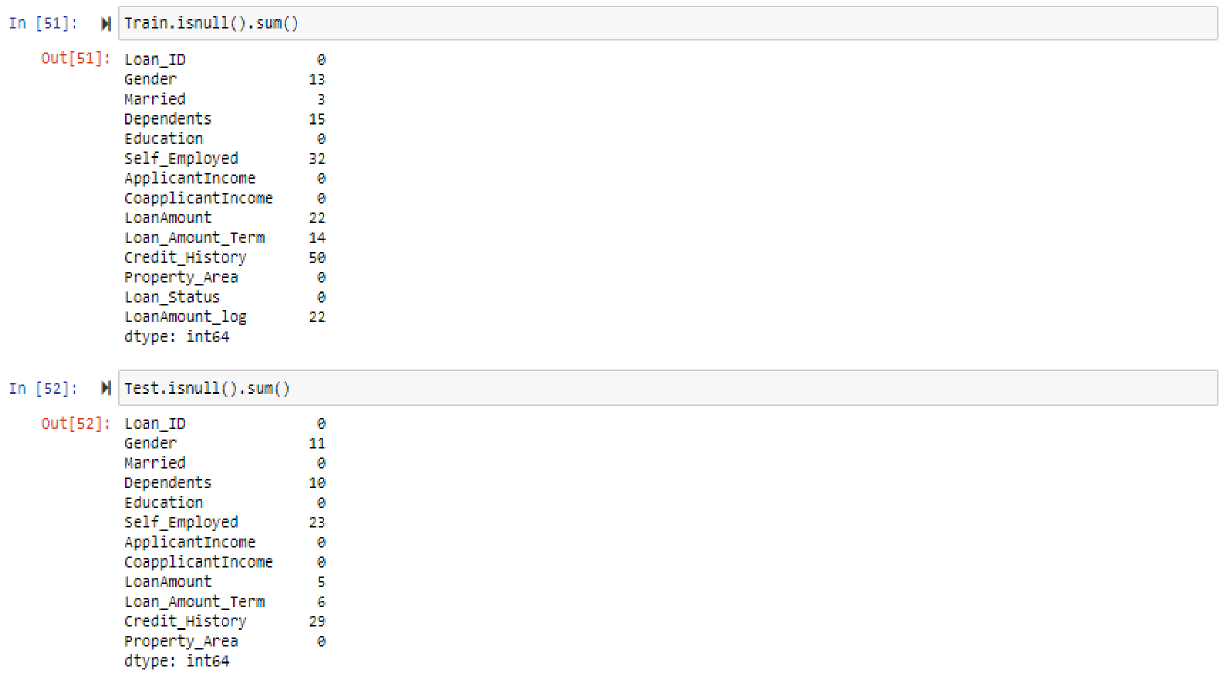
Its a little right skewed so we can try to normalise it.

```
In [50]:  Train['LoanAmount_log']=np.log(Train['LoanAmount'])
          Train['LoanAmount_log'].hist(bins=20)

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x24b6faa3e48>
```



We have applied the log function and also visualize histogram for the log value applied.

It has been normalized.



```
In [51]:  Train.isnull().sum()

Out[51]:  Loan_ID               0
          Gender               13
          Married               3
          Dependents           15
          Education             0
          Self_Employed        32
          ApplicantIncome       0
          CoapplicantIncome     0
          LoanAmount           22
          Loan_Amount_Term     14
          Credit_History       50
          Property_Area         0
          Loan_Status           0
          LoanAmount_log       22
          dtype: int64

In [52]:  Test.isnull().sum()

Out[52]:  Loan_ID               0
          Gender               11
          Married               0
          Dependents           10
          Education             0
          Self_Employed        23
          ApplicantIncome       0
          CoapplicantIncome     0
          LoanAmount            5
          Loan_Amount_Term      6
          Credit_History       29
          Property_Area         0
          dtype: int64
```

Handling Missing Data

These are the Total no.of.missing values are present in each of the variables in the dataset.

In the train dataset, Gender has 13 missing vales, Married has 3, Dependents has 15 missing values and so on

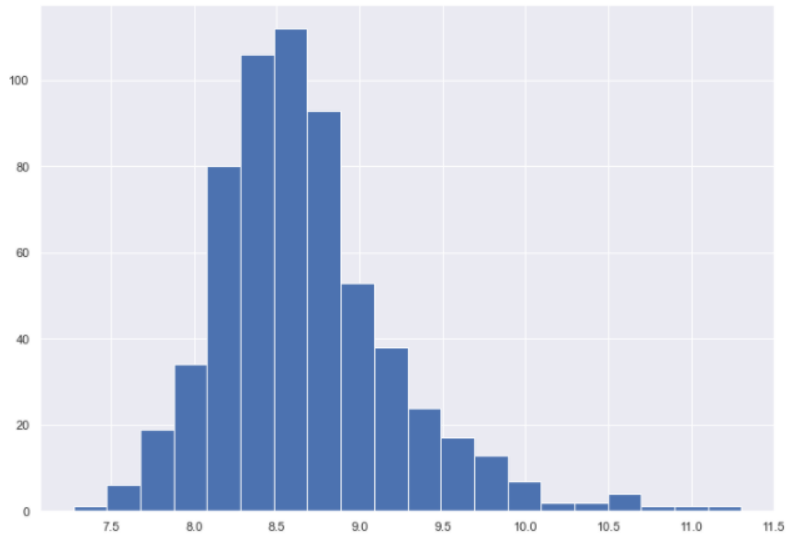
Whereas in the test dataset, Gender has 11 missing values, Married has 0, and dependents has 10.

We can handle these missing vales by filling up these missing values .

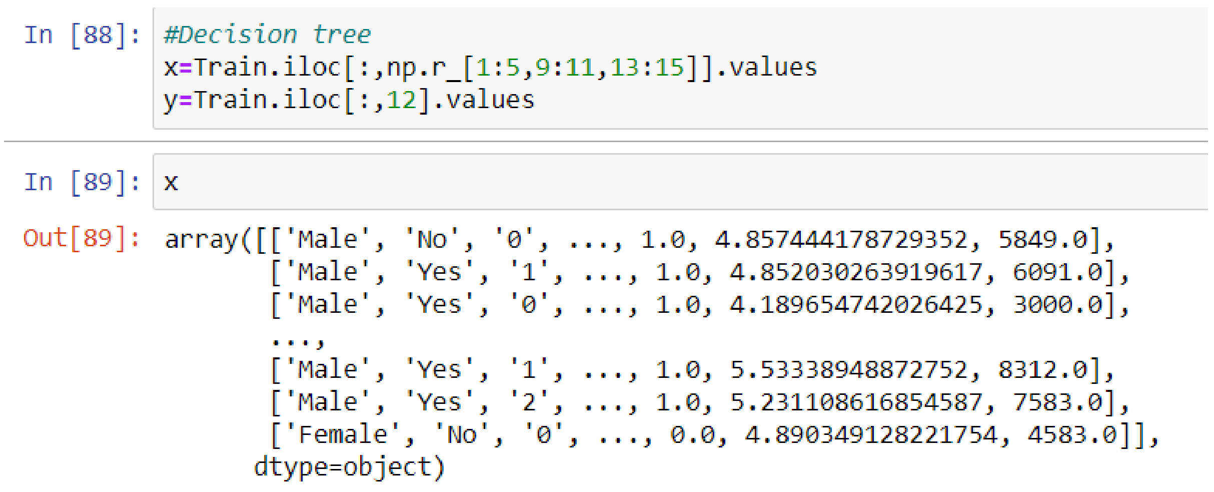
```
In [76]:  Train['Married'].fillna(Train['Married'].mode()[0],inplace=True)

In [77]:  Train['Dependents'].fillna(Train['Dependents'].mode()[0],inplace=True)

In [78]:  Train['Self_Employed'].fillna(Train['Self_Employed'].mode()[0],inplace=True)

In [79]:  Train.LoanAmount=Train.LoanAmount.fillna(Train.LoanAmount.mean())
          Train.LoanAmount_log=Train.LoanAmount_log.fillna(Train.LoanAmount_log.mean())

In [80]:  Train['Loan_Amount_Term'].fillna(Train['Loan_Amount_Term'].mode()[0],inplace=True)

In [81]:  Train['Credit_History'].fillna(Train['Credit_History'].mode()[0],inplace=True)

In [82]:  Train.isnull().sum()

Out[82]:  Loan_ID              0
          Gender               0
          Married              0
          Dependents           0
          Education            0
          Self_Employed        0
          ApplicantIncome      0
          CoapplicantIncome    0
          LoanAmount           0
          Loan_Amount_Term     0
          Credit_History       0
          Property_Area        0
          Loan_Status          0
          LoanAmount_log       0
          dtype: int64
```

We have successfully handled the missing values using mode.

```
In [90]:  Train['TotalIncome_log'].hist(bins=20)

Out[90]:  <matplotlib.axes._subplots.AxesSubplot at 0x24b7035fac8>
```

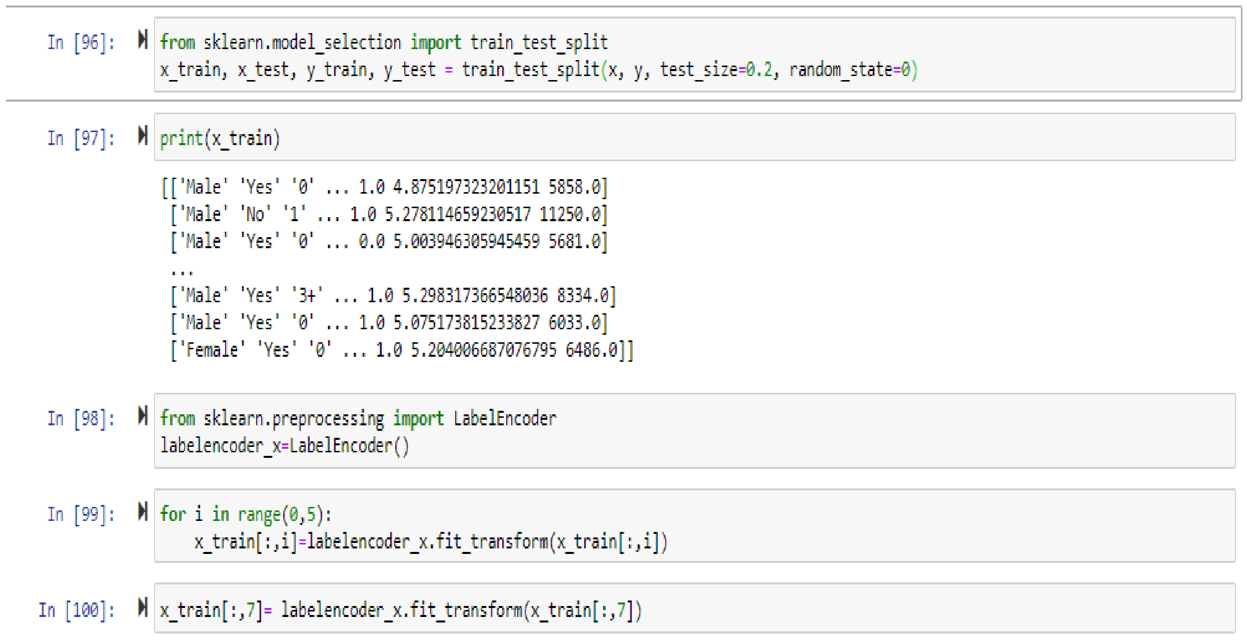


We have visualized the totalamount using histogram.



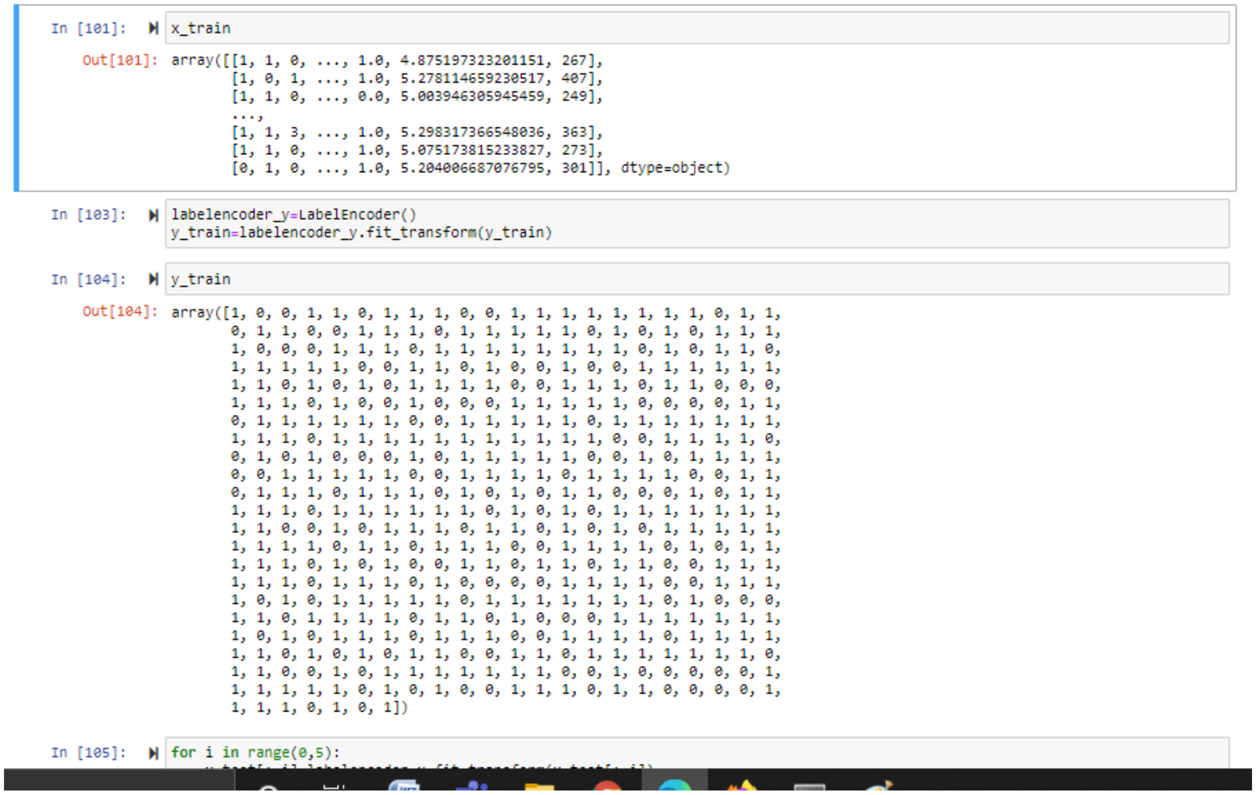
```
In [88]: #Decision tree
         x=Train.iloc[:,np.r_[1:5,9:11,13:15]].values
         y=Train.iloc[:,12].values

In [89]: x

Out[89]: array([['Male', 'No', '0', ..., 1.0, 4.857444178729352, 5849.0],
               ['Male', 'Yes', '1', ..., 1.0, 4.852030263919617, 6091.0],
               ['Male', 'Yes', '0', ..., 1.0, 4.189654742026425, 3000.0],
               ...,
               ['Male', 'Yes', '1', ..., 1.0, 5.53338948872752, 8312.0],
               ['Male', 'Yes', '2', ..., 1.0, 5.231108616854587, 7583.0],
               ['Female', 'No', '0', ..., 0.0, 4.890349128221754, 4583.0]],
              dtype=object)
```

We have separated it as independent and dependent variables. x stores independent variables and y stores dependent variables(only the loan_status is dependent whichis present in the 12th index)



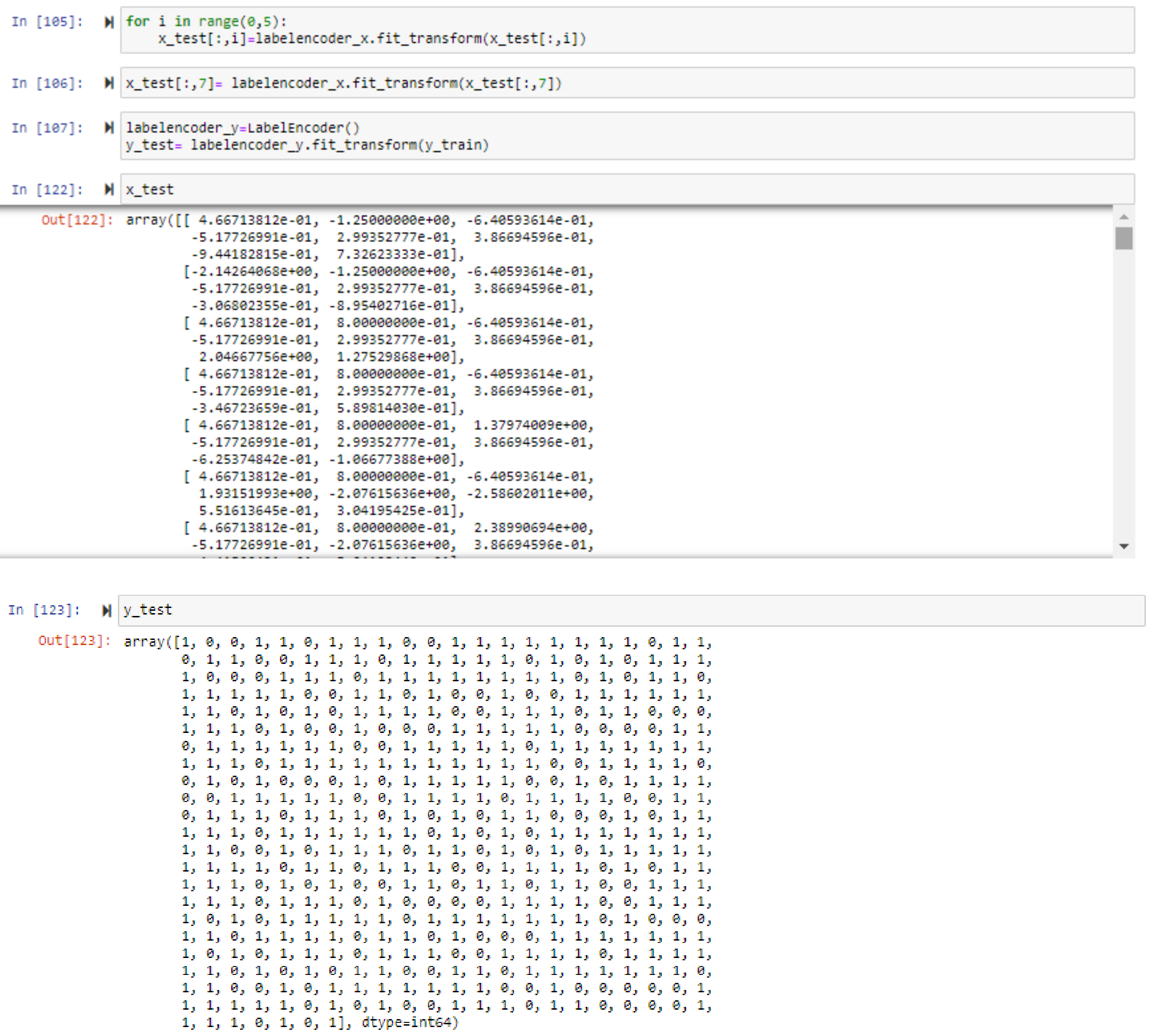
```
In [96]:  from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

In [97]:  print(x_train)

          [['Male' 'Yes' '0' ... 1.0 4.875197323201151 5858.0]
           ['Male' 'No' '1' ... 1.0 5.278114659230517 11250.0]
           ['Male' 'Yes' '0' ... 0.0 5.003946305945459 5681.0]
           ...
           ['Male' 'Yes' '3+' ... 1.0 5.298317366548036 8334.0]
           ['Male' 'Yes' '0' ... 1.0 5.075173815233827 6033.0]
           ['Female' 'Yes' '0' ... 1.0 5.204006687076795 6486.0]]

In [98]:  from sklearn.preprocessing import LabelEncoder
          labelencoder_x=LabelEncoder()

In [99]:  for i in range(0,5):
              x_train[:,i]=labelencoder_x.fit_transform(x_train[:,i])

In [100]: x_train[:,7]= labelencoder_x.fit_transform(x_train[:,7])
```

we have now split our dataset into x_test, x_train and y_train,y_test using sklearn x_test, x_train will have the independent variables and y_test, y_train will have the dependent ones we have split it in the ratio of 80:20 where 80% of it is training and 20% of it is testing we have used label encoder to convert everything to 0s and 1s so that the machine can understand we have applied 'for loop' for every index that needs to be converted.



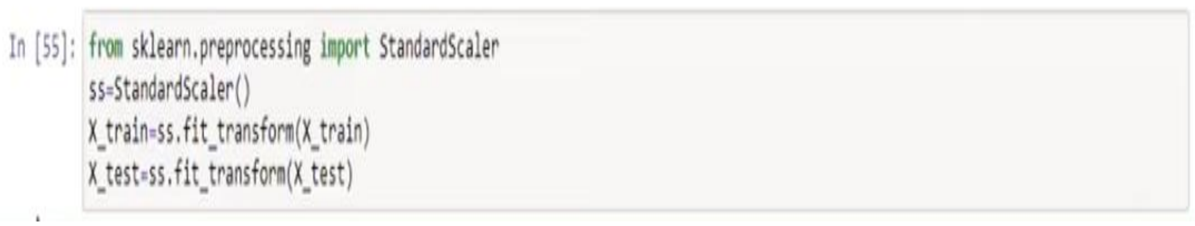
```
In [101]:  x_train

Out[101]: array([[1, 1, 0, ..., 1.0, 4.875197323201151, 267],
                 [1, 0, 1, ..., 1.0, 5.278114659230517, 407],
                 [1, 1, 0, ..., 0.0, 5.003946305945459, 249],
                 ...,
                 [1, 1, 3, ..., 1.0, 5.298317366548036, 363],
                 [1, 1, 0, ..., 1.0, 5.075173815233827, 273],
                 [0, 1, 0, ..., 1.0, 5.204006687076795, 301]], dtype=object)

In [103]:  labelencoder_y=LabelEncoder()
           y_train=labelencoder_y.fit_transform(y_train)

In [104]:  y_train

Out[104]: array([1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
                 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
                 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
                 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
                 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
                 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
                 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
                 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
                 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
                 1, 1, 1, 0, 1, 0, 1])

In [105]:  for i in range(0,5):
```

Both test and train has been converted to numeric values.



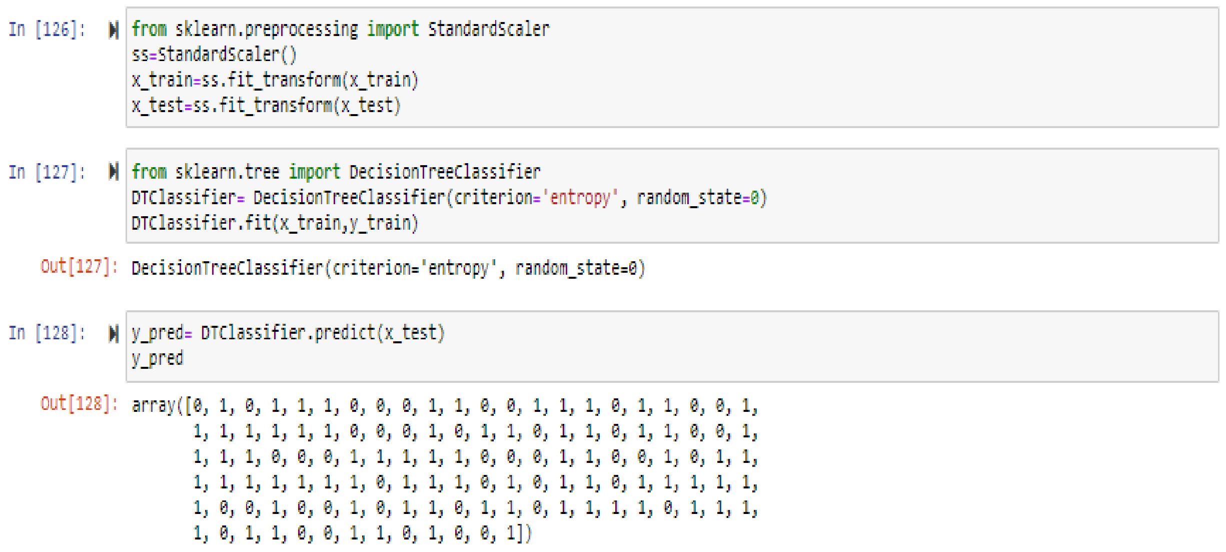
```
In [105]:  for i in range(0,5):
               x_test[:,i]=labelencoder_x.fit_transform(x_test[:,i])

In [106]:  x_test[:,7]= labelencoder_x.fit_transform(x_test[:,7])

In [107]:  labelencoder_y=LabelEncoder()
           y_test= labelencoder_y.fit_transform(y_train)

In [122]:  x_test

Out[122]: array([[ 4.66713812e-01, -1.25000000e+00, -6.40593614e-01,
                  -5.17726991e-01,  2.99352777e-01,  3.86694596e-01,
                  -9.44182815e-01,  7.32623333e-01],
                 [-2.14264068e+00, -1.25000000e+00, -6.40593614e-01,
                  -5.17726991e-01,  2.99352777e-01,  3.86694596e-01,
                  -3.06802355e-01, -8.95402716e-01],
                 [ 4.66713812e-01,  8.00000000e-01, -6.40593614e-01,
                  -5.17726991e-01,  2.99352777e-01,  3.86694596e-01,
                   2.04667756e+00,  1.27529868e+00],
                 [ 4.66713812e-01,  8.00000000e-01, -6.40593614e-01,
                  -5.17726991e-01,  2.99352777e-01,  3.86694596e-01,
                  -3.46723659e-01,  5.89814030e-01],
                 [ 4.66713812e-01,  8.00000000e-01,  1.37974009e+00,
                  -5.17726991e-01,  2.99352777e-01,  3.86694596e-01,
                  -6.25374842e-01, -1.06677388e+00],
                 [ 4.66713812e-01,  8.00000000e-01, -6.40593614e-01,
                   1.93151993e+00, -2.07615636e+00, -2.58602011e+00,
                   5.51613645e-01,  3.04195425e-01],
                 [ 4.66713812e-01,  8.00000000e-01,  2.38990694e+00,
                  -5.17726991e-01, -2.07615636e+00,  3.86694596e-01,

In [123]:  y_test

Out[123]: array([1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
                 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
                 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
                 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
                 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
                 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
                 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
                 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
                 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
                 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
                 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

We have converted it similarly for both the test datasets



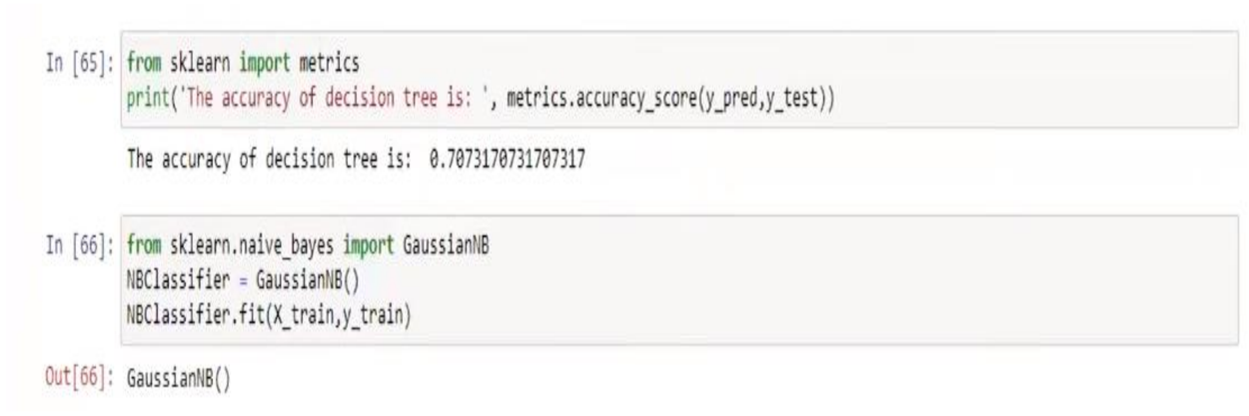
```
In [55]: from sklearn.preprocessing import StandardScaler
         ss=StandardScaler()
         X_train=ss.fit_transform(X_train)
         X_test=ss.fit_transform(X_test)
```

we have scaled our dataset



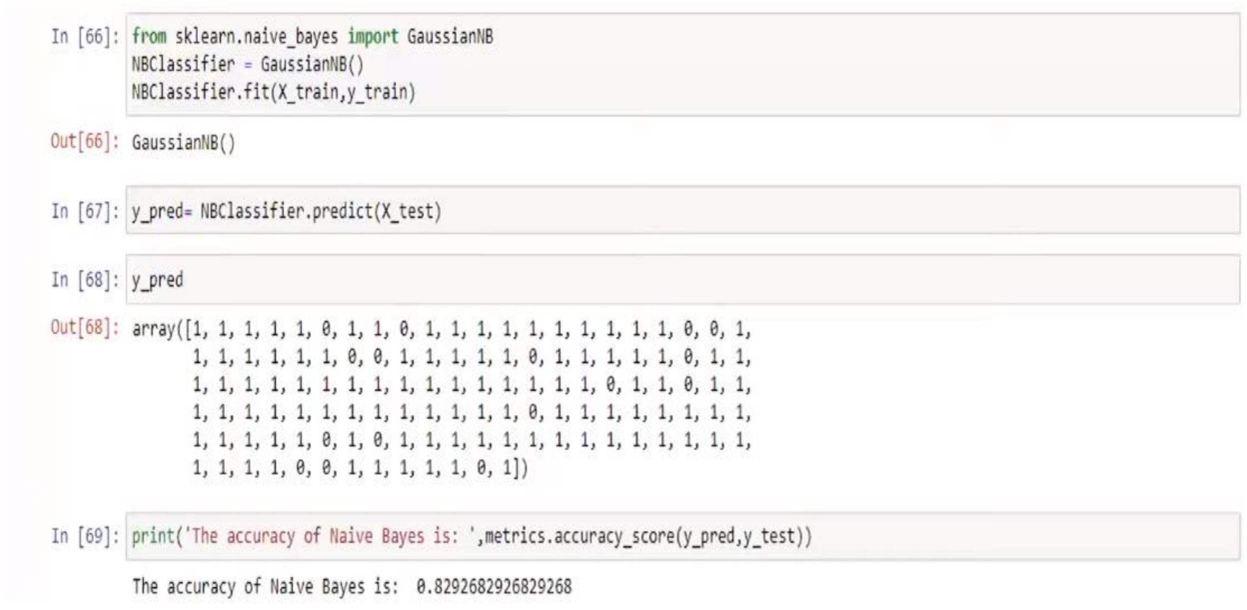
```
In [126]:  from sklearn.preprocessing import StandardScaler
           ss=StandardScaler()
           x_train=ss.fit_transform(x_train)
           x_test=ss.fit_transform(x_test)

In [127]:  from sklearn.tree import DecisionTreeClassifier
           DTClassifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
           DTClassifier.fit(x_train,y_train)

Out[127]:  DecisionTreeClassifier(criterion='entropy', random_state=0)

In [128]:  y_pred= DTClassifier.predict(x_test)
           y_pred

Out[128]:  array([0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
                  1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
                  1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
                  1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
                  1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
                  1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1])
```

After scaling the dataset we have tried apply decisionTree Algorithm to our dataset and predict the accuracy of the outcome



```
In [65]:  from sklearn import metrics
          print('The accuracy of decision tree is: ', metrics.accuracy_score(y_pred,y_test))

          The accuracy of decision tree is:  0.7073170731707317

In [66]:  from sklearn.naive_bayes import GaussianNB
          NBClassifier = GaussianNB()
          NBClassifier.fit(X_train,y_train)

Out[66]:  GaussianNB()
```

After predicting the accuracy we find that its not so accurate. It shows only 70% accuracy. So we have tried to import another algorithm called naive_bayes.

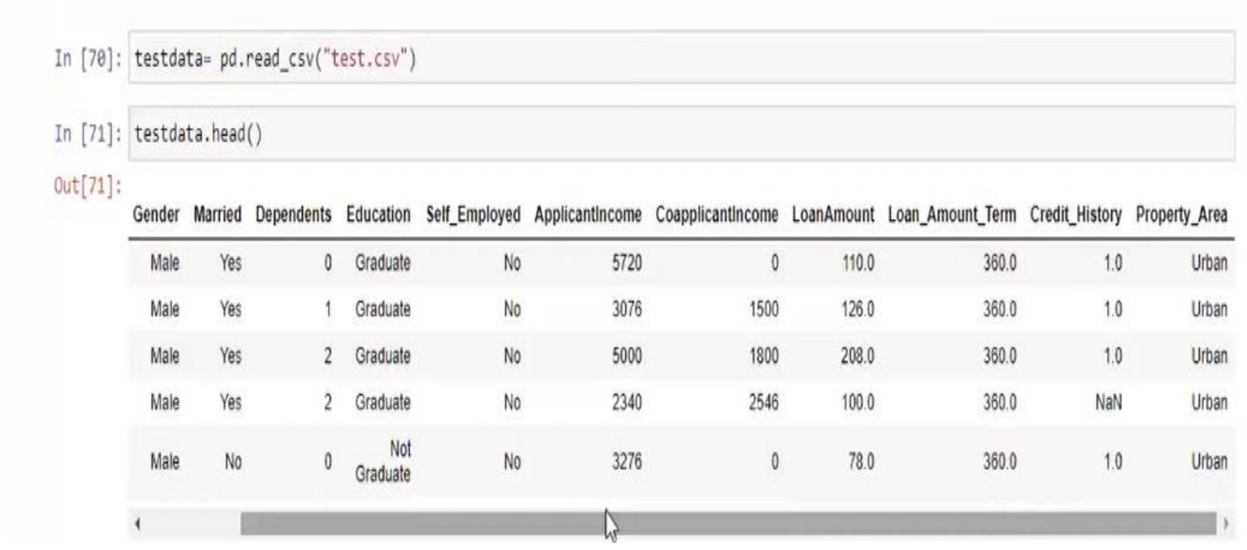


```
In [66]: from sklearn.naive_bayes import GaussianNB
         NBClassifier = GaussianNB()
         NBClassifier.fit(X_train,y_train)

Out[66]: GaussianNB()

In [67]: y_pred= NBClassifier.predict(X_test)

In [68]: y_pred

Out[68]: array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
                1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1])

In [69]: print('The accuracy of Naive Bayes is: ',metrics.accuracy_score(y_pred,y_test))

         The accuracy of Naive Bayes is:  0.8292682926829268
```

It shows almost 83% accuracy which is much better than the DecisionTree.

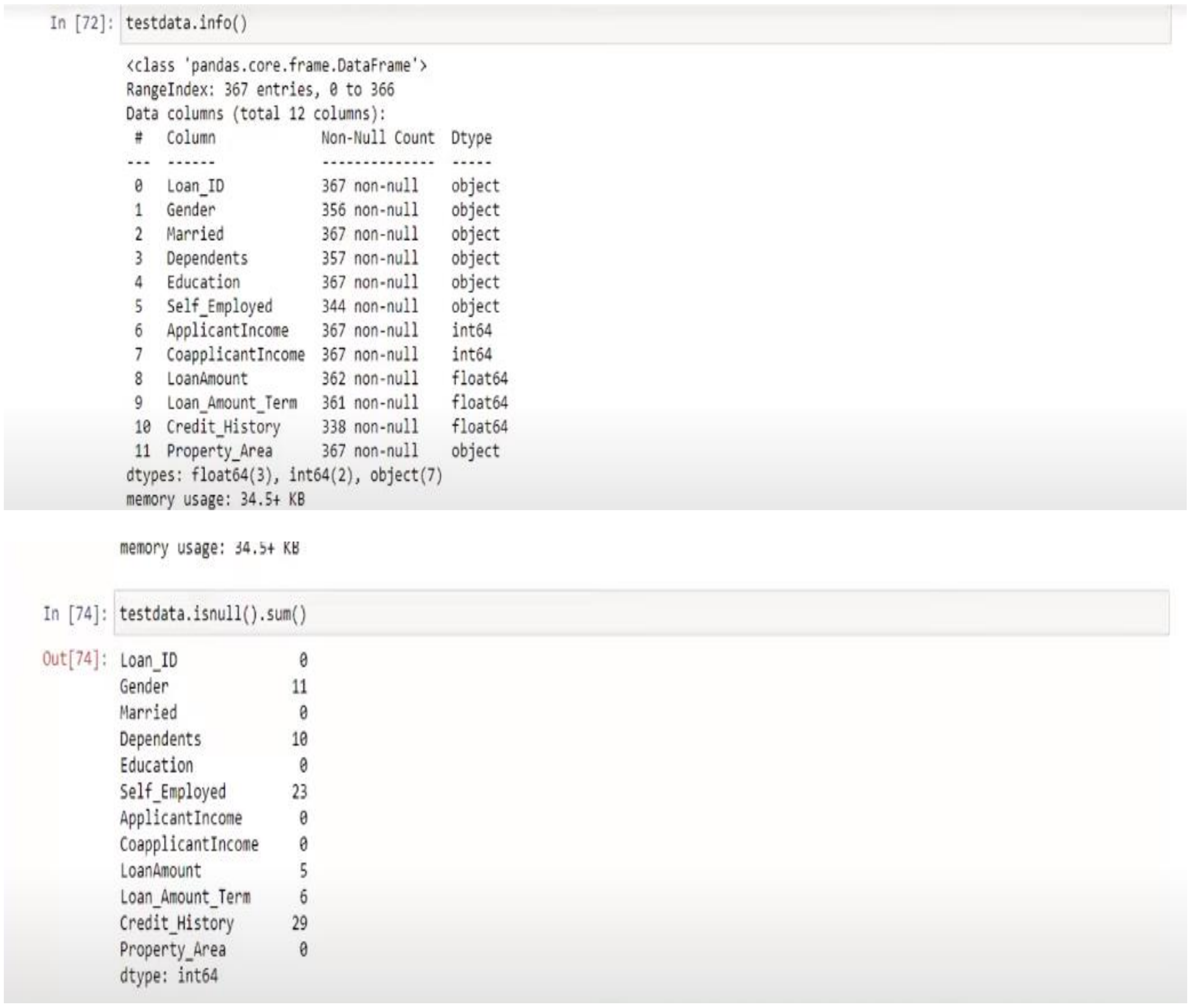
Naive_Bayes:

We are going to use this algorithm to predict the outcome of our data.



```
In [70]: testdata= pd.read_csv("test.csv")

In [71]: testdata.head()

Out[71]:
```

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

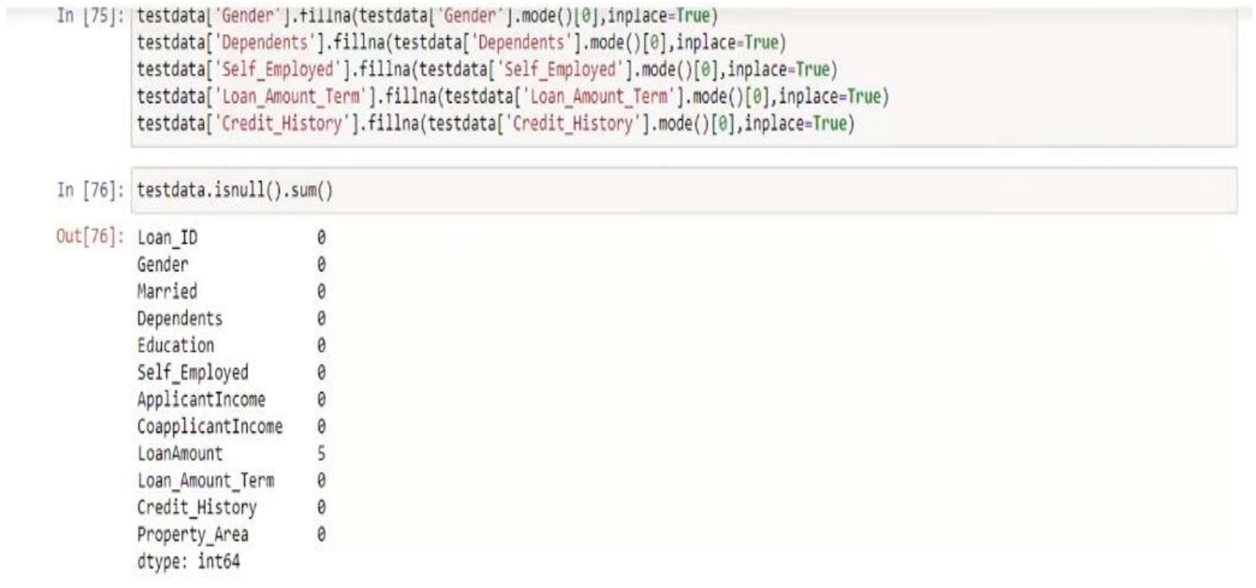


```
In [72]: testdata.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

```
memory usage: 34.5+ KB
```

```
In [74]: testdata.isnull().sum()

Out[74]: Loan_ID              0
         Gender              11
         Married              0
         Dependents          10
         Education            0
         Self_Employed       23
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount           5
         Loan_Amount_Term     6
         Credit_History      29
         Property_Area        0
         dtype: int64
```

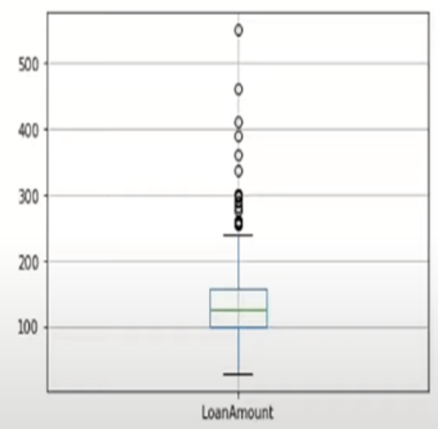
The values predicted are almost the same since this also shows some missing values we are about to handle them.

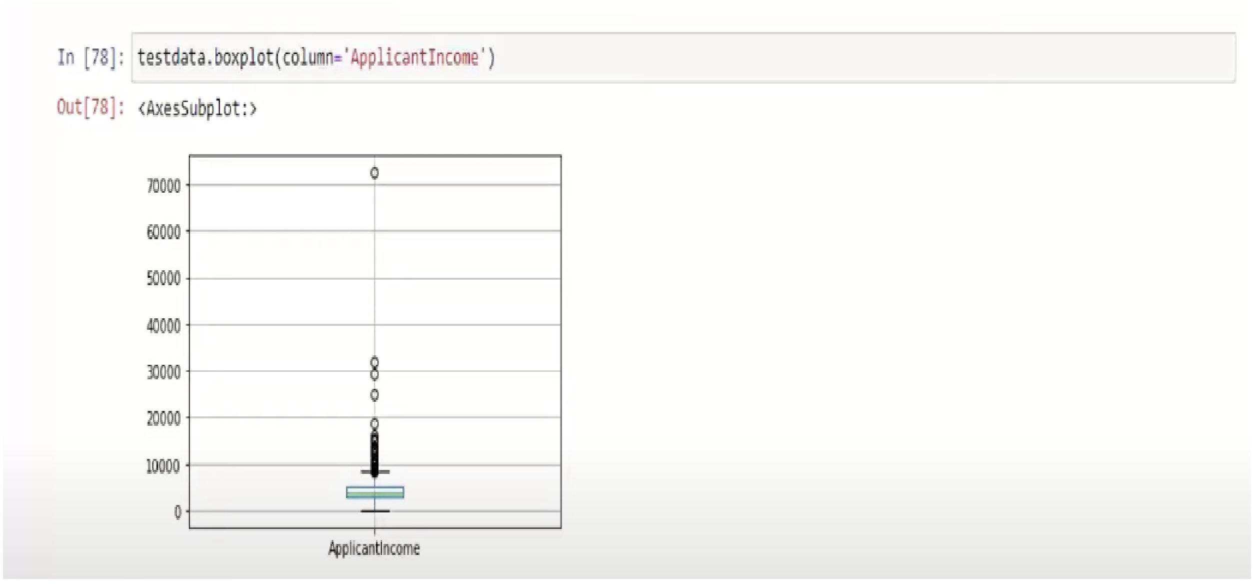


```
In [75]: testdata['Gender'].fillna(testdata['Gender'].mode()[0],inplace=True)
         testdata['Dependents'].fillna(testdata['Dependents'].mode()[0],inplace=True)
         testdata['Self_Employed'].fillna(testdata['Self_Employed'].mode()[0],inplace=True)
         testdata['Loan_Amount_Term'].fillna(testdata['Loan_Amount_Term'].mode()[0],inplace=True)
         testdata['Credit_History'].fillna(testdata['Credit_History'].mode()[0],inplace=True)

In [76]: testdata.isnull().sum()

Out[76]: Loan_ID              0
         Gender               0
         Married              0
         Dependents           0
         Education            0
         Self_Employed        0
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount           5
         Loan_Amount_Term     0
         Credit_History       0
         Property_Area        0
         dtype: int64
```

LoanAmount is not yet handled but, we are just going to check for the outliers.

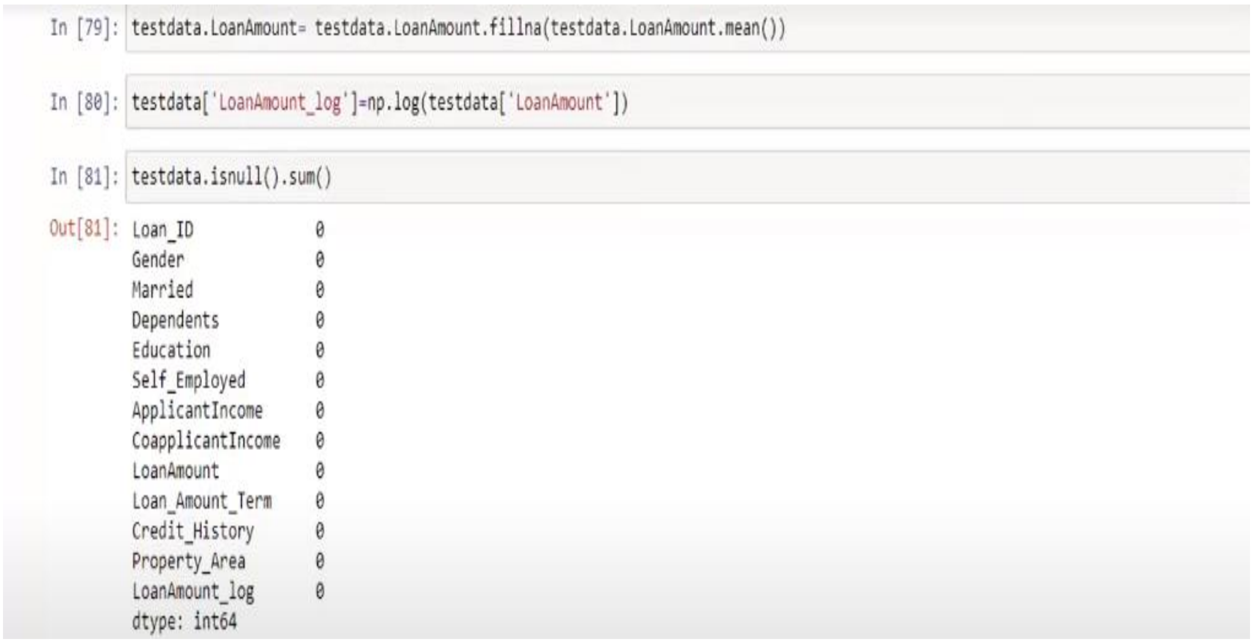
```
In [77]: testdata.boxplot(column='LoanAmount')

Out[77]: <AxesSubplot:>
```



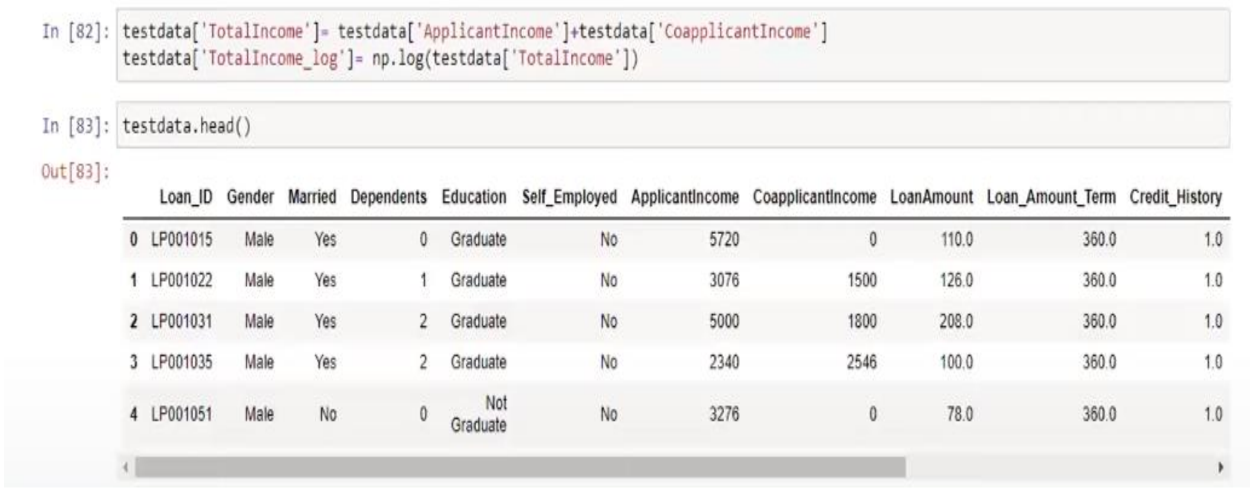


```
In [78]: testdata.boxplot(column='ApplicantIncome')

Out[78]: <AxesSubplot:>
```

It has a lot of outliers,We are going to handle these outliers now.



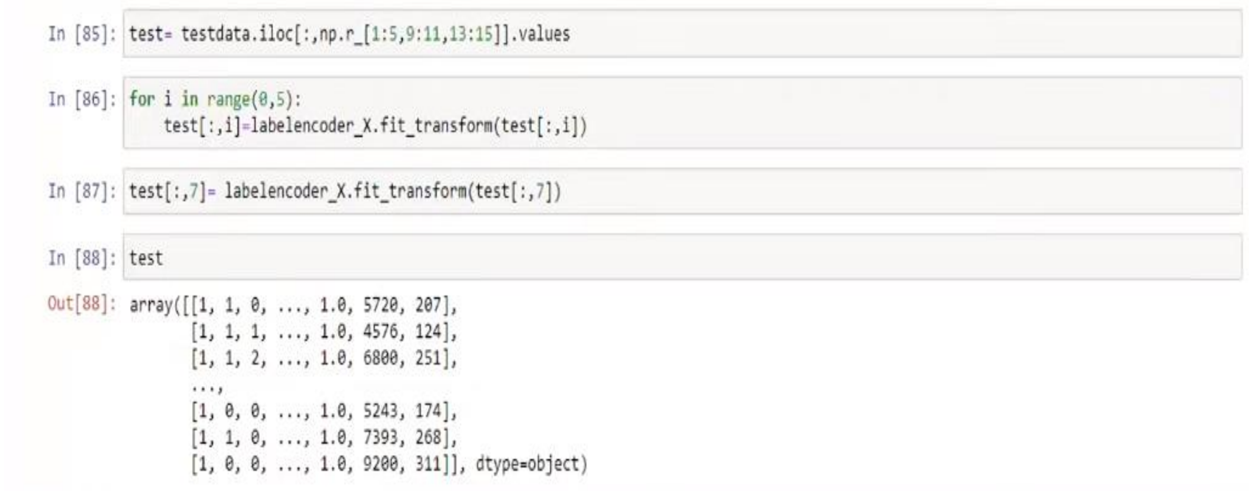
```
In [79]: testdata.LoanAmount= testdata.LoanAmount.fillna(testdata.LoanAmount.mean())

In [80]: testdata['LoanAmount_log']=np.log(testdata['LoanAmount'])

In [81]: testdata.isnull().sum()

Out[81]: Loan_ID              0
         Gender               0
         Married              0
         Dependents           0
         Education            0
         Self_Employed        0
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount           0
         Loan_Amount_Term     0
         Credit_History       0
         Property_Area        0
         LoanAmount_log       0
         dtype: int64
```



```
In [82]: testdata['TotalIncome']= testdata['ApplicantIncome']+testdata['CoapplicantIncome']
         testdata['TotalIncome_log']= np.log(testdata['TotalIncome'])

In [83]: testdata.head()

Out[83]:
```

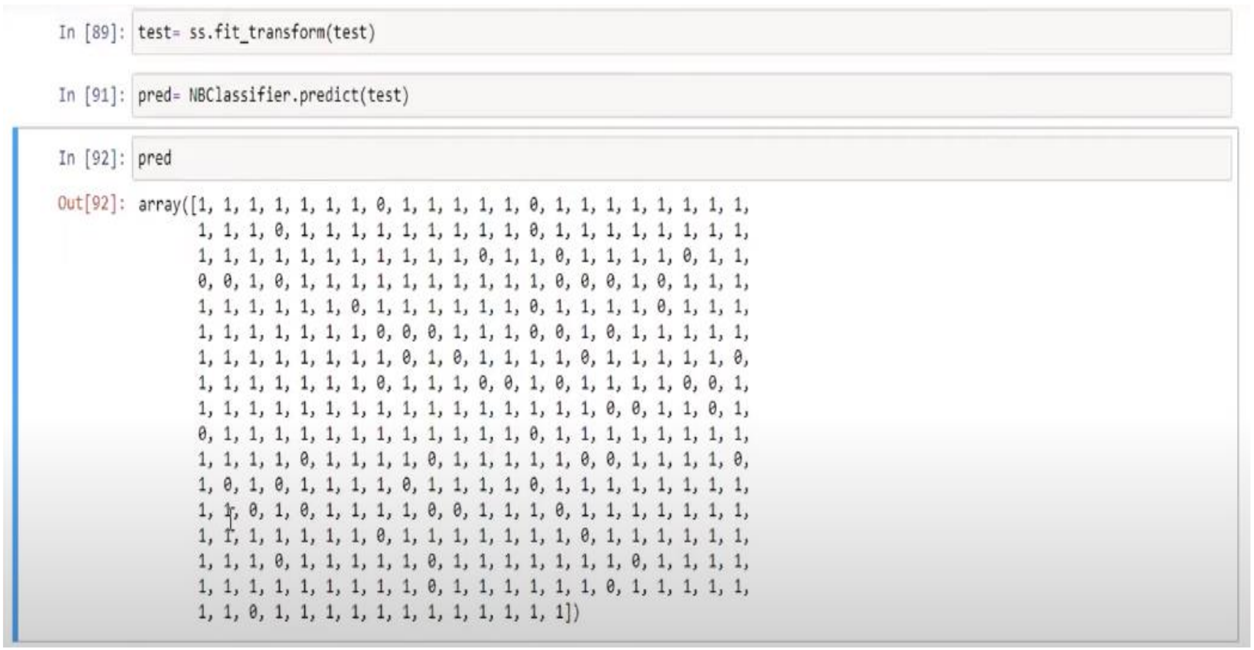
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	1.0
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0

We've tried to normalize our data here.



```
In [85]: test= testdata.iloc[:,np.r_[1:5,9:11,13:15]].values

In [86]: for i in range(0,5):
             test[:,i]=labelencoder_X.fit_transform(test[:,i])

In [87]: test[:,7]= labelencoder_X.fit_transform(test[:,7])

In [88]: test

Out[88]: array([[1, 1, 0, ..., 1.0, 5720, 207],
                [1, 1, 1, ..., 1.0, 4576, 124],
                [1, 1, 2, ..., 1.0, 6800, 251],
                ...,
                [1, 0, 0, ..., 1.0, 5243, 174],
                [1, 1, 0, ..., 1.0, 7393, 268],
                [1, 0, 0, ..., 1.0, 9200, 311]], dtype=object)
```

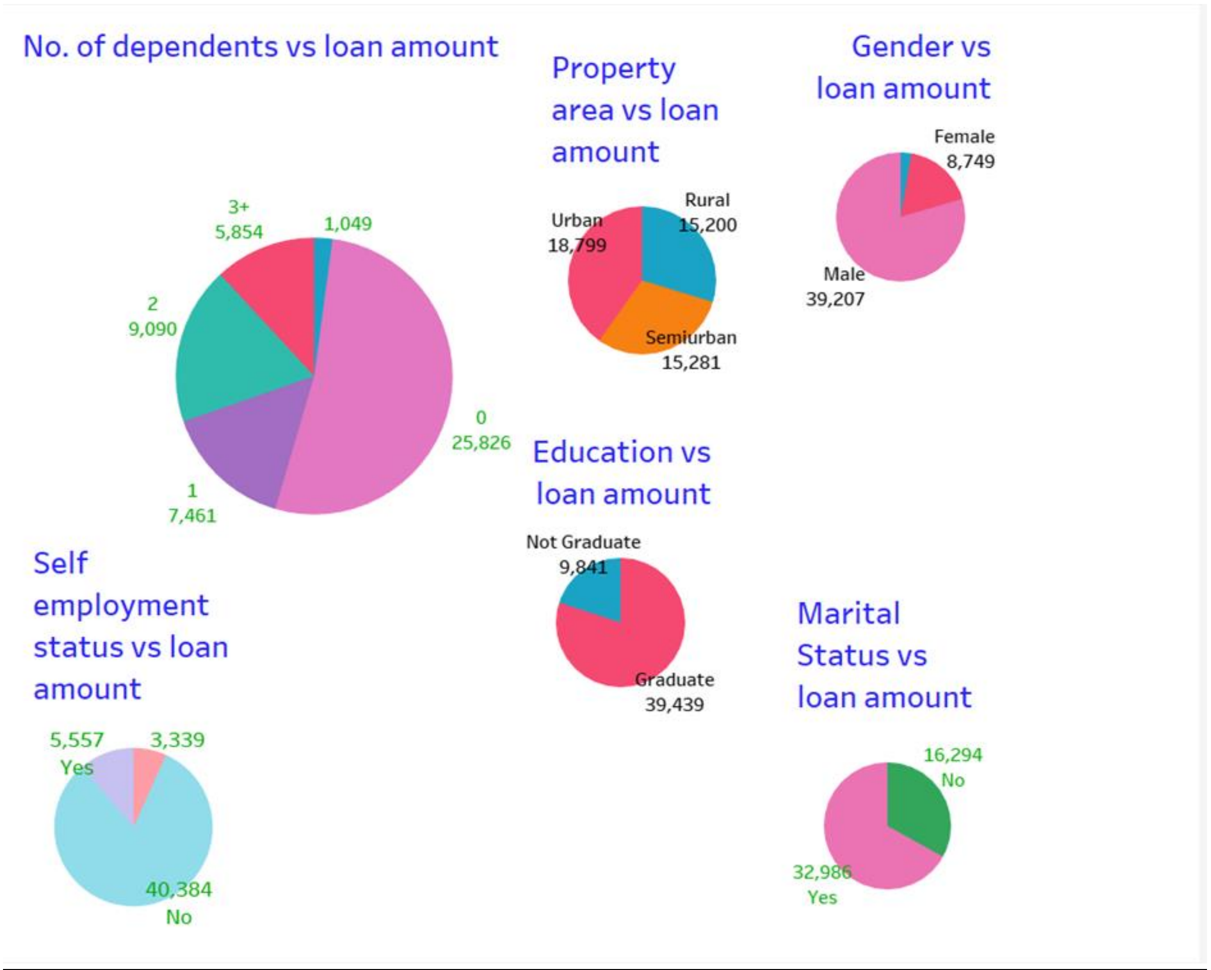
As we have done before we have tried to change the categorical data to numerical data. We are going to scale our data now. Finally we are going to predict whether or not the person is eligible for a loan based on this instance of NBClassifier we've created before.



```
In [89]: test= ss.fit_transform(test)

In [91]: pred= NBClassifier.predict(test)

In [92]: pred

Out[92]: array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
                0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
                0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
                1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

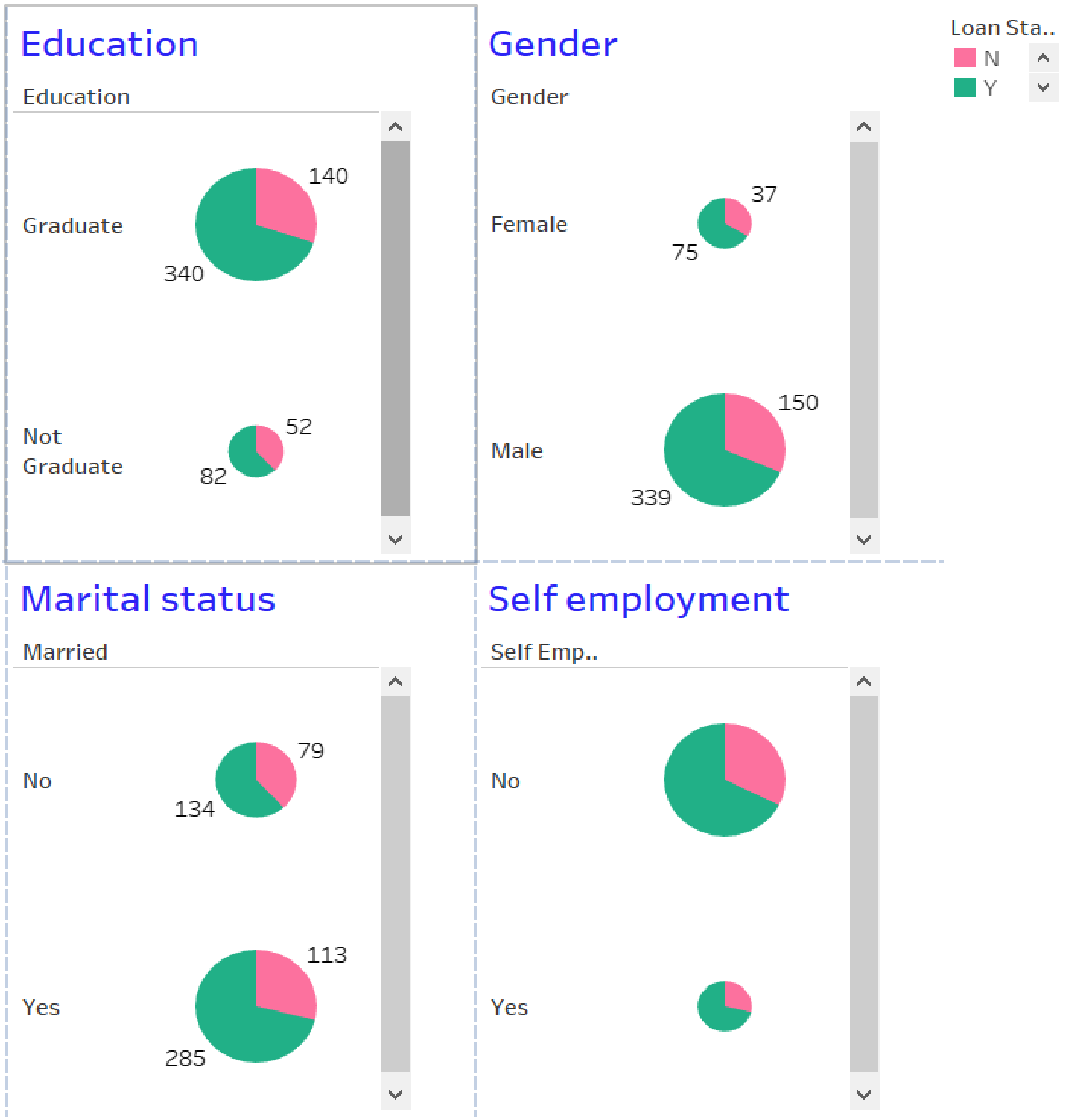
Here 1 indicates that a person is eligible and 0 indicates that a person is not eligible.

USING TABLEAU:

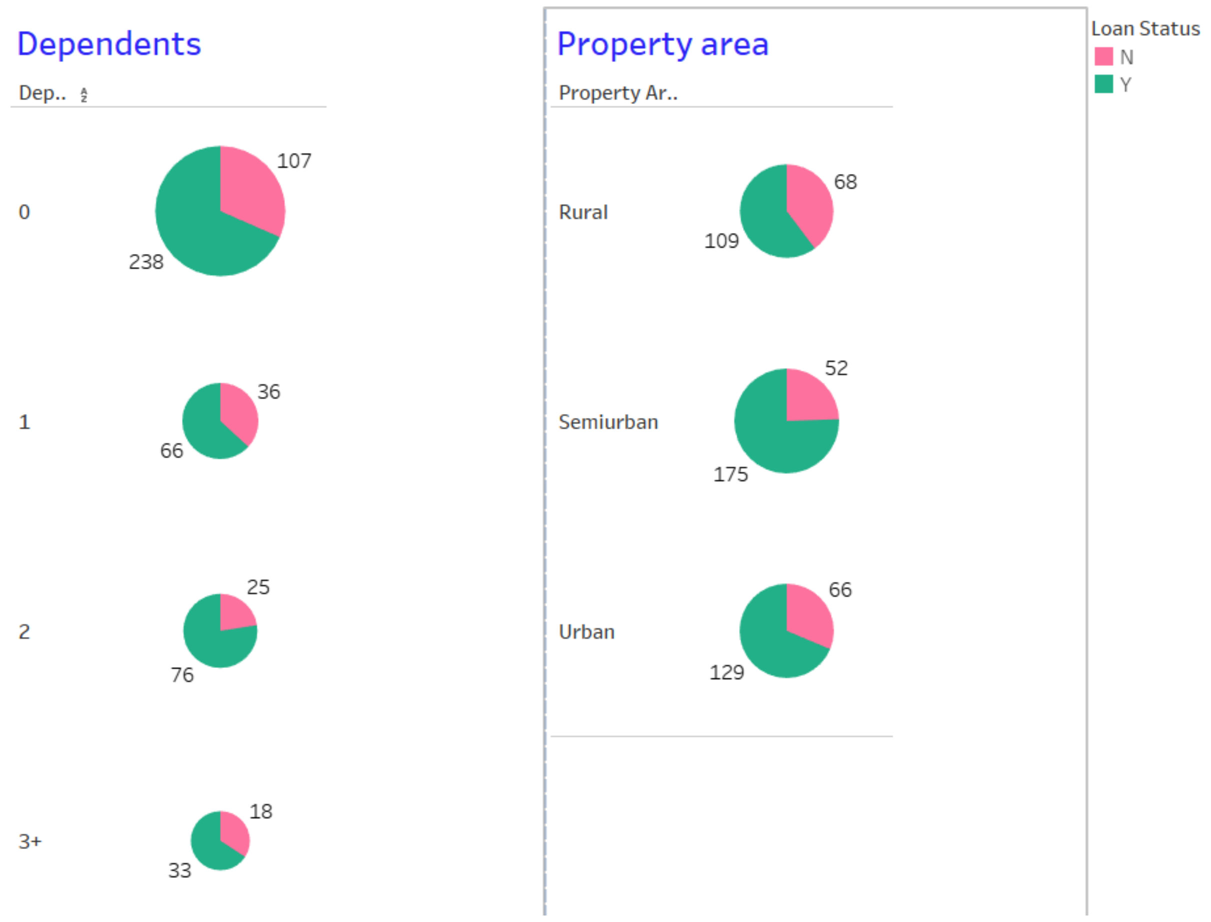


- People with no dependants have applied for more loans.
- More males have applied for a loan than females.
- Urban area dwellers have applied the most but semi urban and rural settlers applicants don’t differ by a large quantity.

- Self-employed people have applied for 1/8th of the loans non self-employed people have applied for.
- Graduates are 3 times more likely to apply for a loan.
- More married people have applied for loans.

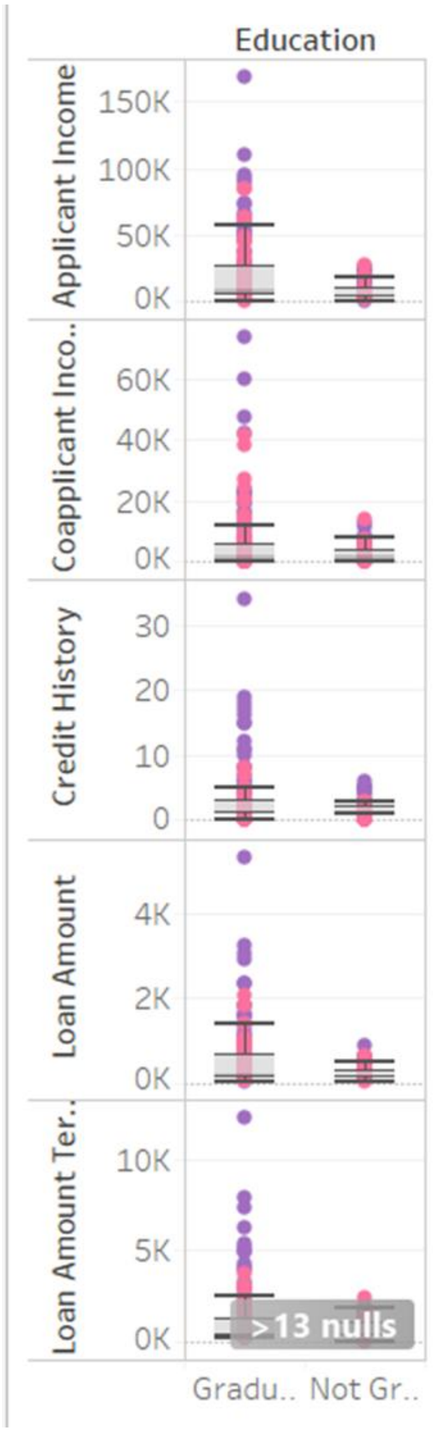


- Graduates are more than twice as likely to get their loan approved.
- Both genders have an equal chance of getting their loan approved.
- Married couples have had more loans approved.
- Self-employment statuses don't differ much when it comes to loan approval chances.



- Loans applied by applicants with 2 dependents seem to be approved more.

- Loans applied by applicants with 1, 3+ have the same chance of being approved.
- Semi urban applicants have more loans approved than rural and urban applicants.



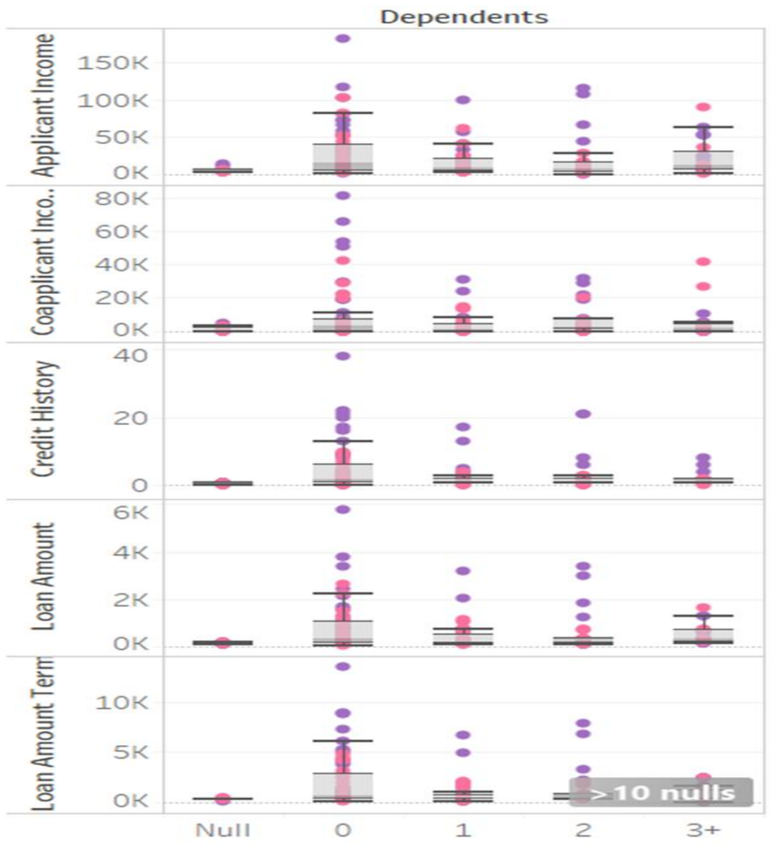
- Outlier loans approved over upper whisker 57102 when comparing education and applicant income are mostly approved but rejected cases show a credit history of 0.
- Chances of getting a loan approved is very low if the applicant is not a graduate.
- Loans applied by graduates are most likely to be approved only when the applicant income exceeds the upper whisker value.

Comparing education with co-applicant income

- It is almost impossible to get a loan approved for non-graduates unless co-applicant income is higher than expected and credit history is 2.
- Only loans applied by graduates with a co-applicant whose income exceeds 40k have been approved
- Comparing credit history with education.
- Only graduates whose sum(credit history)>10 have loans approved.
- Non graduate applicants have a slim chance of loan approval only if sum(credit history)>5

Comparing loan amount and education

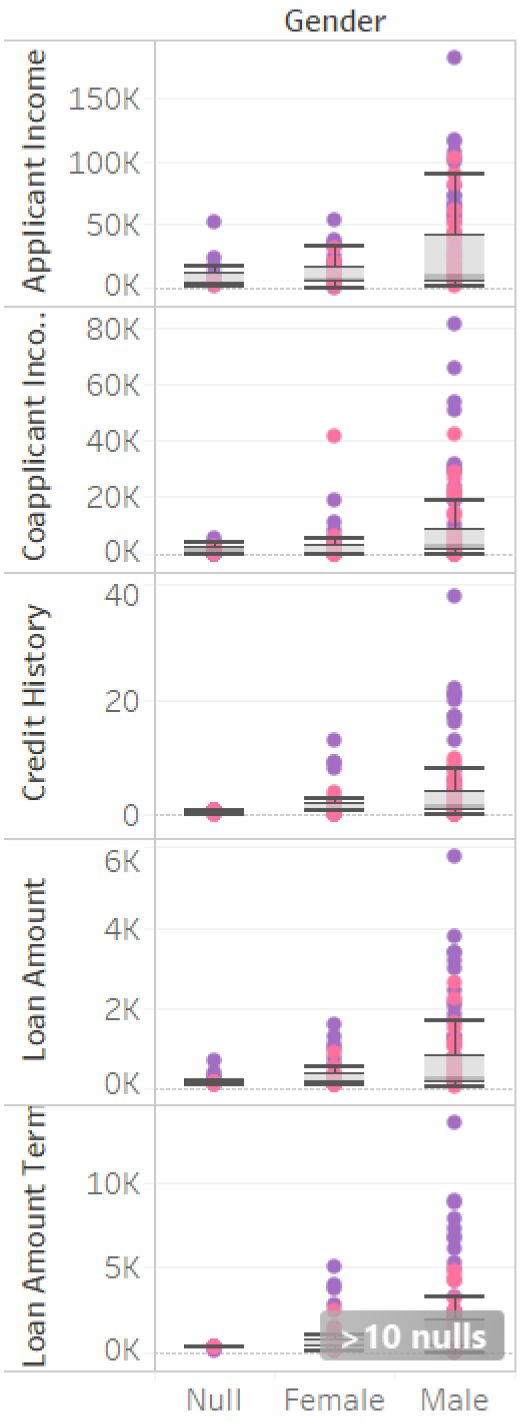
- Graduates whose loan amount exceeds 2k and/ credit history>12 have loans approved.
- Non-graduates have loans approved if their credit history exceeds 5.
- Comparing loan amount term and education.
- Graduates with credit history of 10 and more have loans approved.
- There isn't enough data about non-graduate applicants to make an inference.



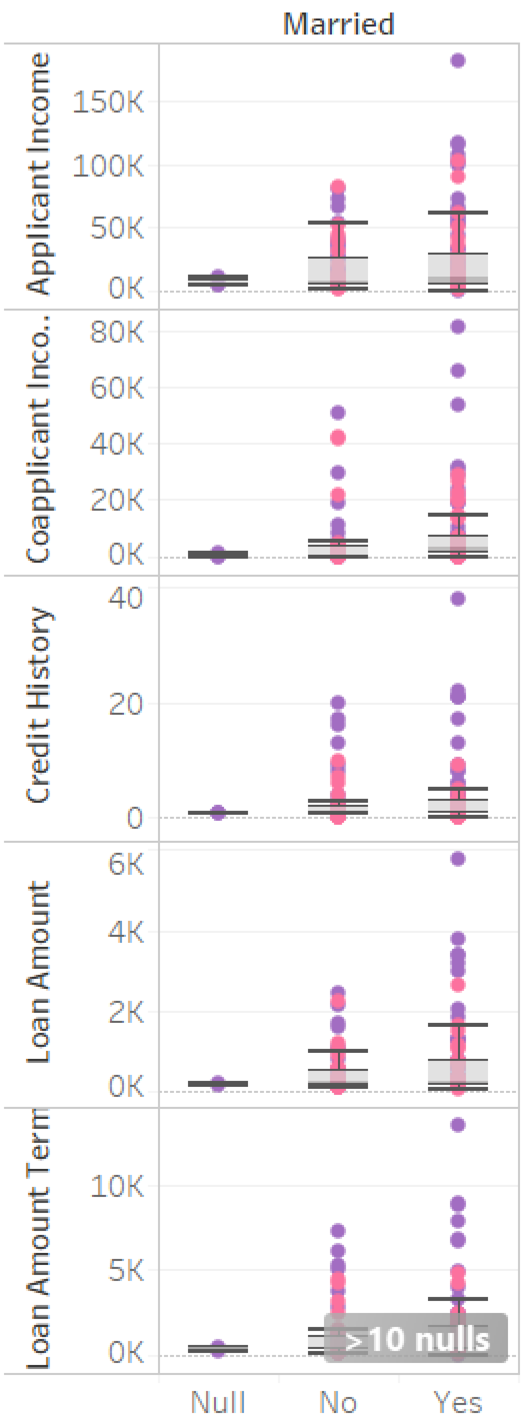
- When comparing dependants and applicant income, Applicants with 2 dependants' loans get approved. when Applicant income exceeds upper whisker value Applicants with lower credit history is likely to get disapproved even if applicant income is above average.

- Credit history is more important than applicant income irrespective of the number of dependants.
- When comparing co-applicant income with dependents, Having co-applicant with income above 50k ensures loan approval when there is just one dependent.
- Co-applicant income above 20k ensures loan approval when number of dependents is 2.
- Credit score especially matters when number of dependents is 2+
- When comparing credit history and dependents Applicants whose credit history is above average are most likely to have loans approved.
- When comparing loan amount with dependents Only credit history of 8 and above helps.
- When comparing loan amount term and dependents Applicants with 2+ dependents have more than 10 null values so inference might not be accurate.

- A rural female applicant's loan has been approved over male counterparts with the same credit history of 8.



- When comparing Gender with applicant income Males with credit history above 20 are most likely to have their loans approved even if their income isn't above expected range.
- Only credit history above 8 seems to guarantee the loan approval of a female applicant.
- When comparing co-applicant income with gender, A female with 3+ dependents and no discernable credit history has had her loan application rejected.
- Females with credit history above 9 have loans approved.
- When comparing loan amount with gender,
  A male with a credit history of 6 has had their load application approved only because of their income of 65k and their co-applicant's income of 21k.
- Females with a credit history of 8 and above have their loans approved.
- When comparing loan amount term with gender, Above 10 null values might hinder accuracy of inferences.
- Credit history above 9 is required for the female loan approvals.
- Credit history above 16 is required for male applicant's loan approvals.



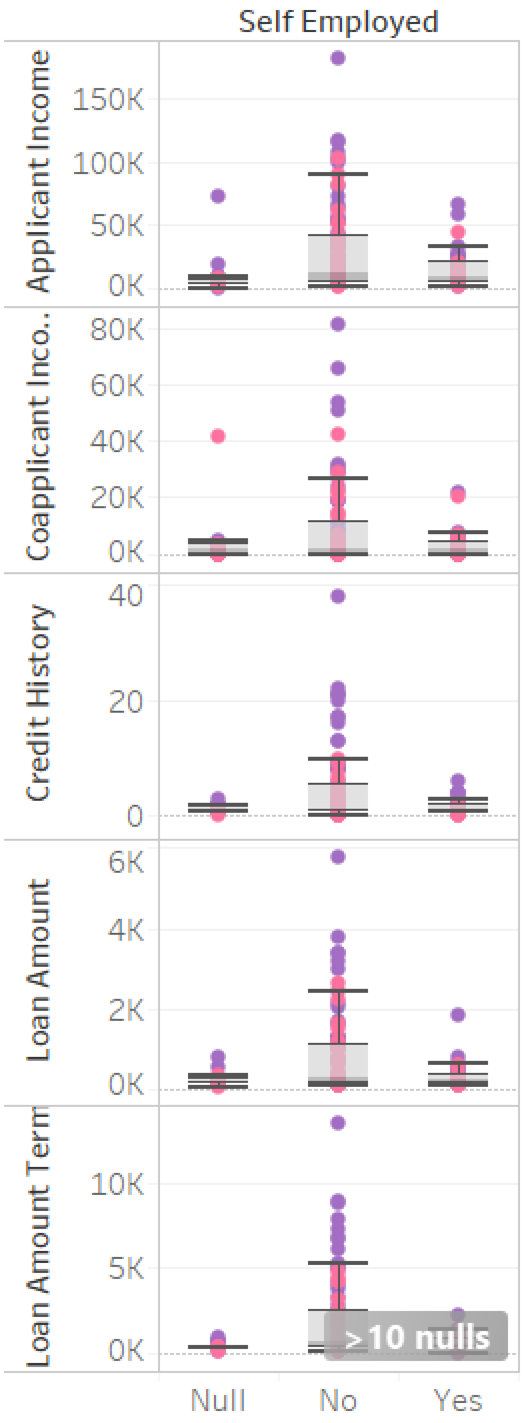
- When comparing applicant income with marital status,A married person’s loan has been approved despite a credit history of 2 only because the incomes of both the applicant and the co-applicant are above average.

- A credit history of 17 and above is necessary for a married person’s loan approval.
- A credit history of 16 and above is necessary for an unmarried person’s loan approval having a higher income does not guarantee approval.
- When comparing co-applicant incomes with marital status,Only credit history of 21 and above guarantees loan approval in a married person’s case.
- Credit history of 8 is enough if co-applicants income exceeds expectations and applicant is an unmarried female.
- When comparing credit history with marital status, Credit history of 9 is enough if applicant is an unmarried female.
- An unmarried male with a credit score of 13 and above will be granted a loan.
- When comparing loan amount with marital status, A married male’s loan has been disapproved despite having no dependents and a credit history of 9 despite having a higher income than his counterparts (reason unknown)

- An unmarried male with a credit history of 17 has been turned done for unknown reasons as similar applicants’ loans have been approved.
- When comparing loan amount term with marital status,10+ nulls hinder accuracy of inferences.
- Unmarried female with credit history of 8 has received loan approval but unmarried males require credit history of 13.
- Married male with a credit history of just 8 and 2 dependents has had his loan application approved over eligible counterparts (reason unknown).
- Married males need a credit history of 21 and above for loan approval.
- Unmarried male applicant loans are approved if they have a credit history of 20 and above.
- When comparing applicant income with property area,Rural male applicant loans are approved if they have a credit history of 6 or 8 and dependents .
- It is very rare for a rural female to receive a loan and even applicants with higher credit histories and incomes have been turned down .
- Credit history of 20 and above matters in the case of semi-urban applicants.

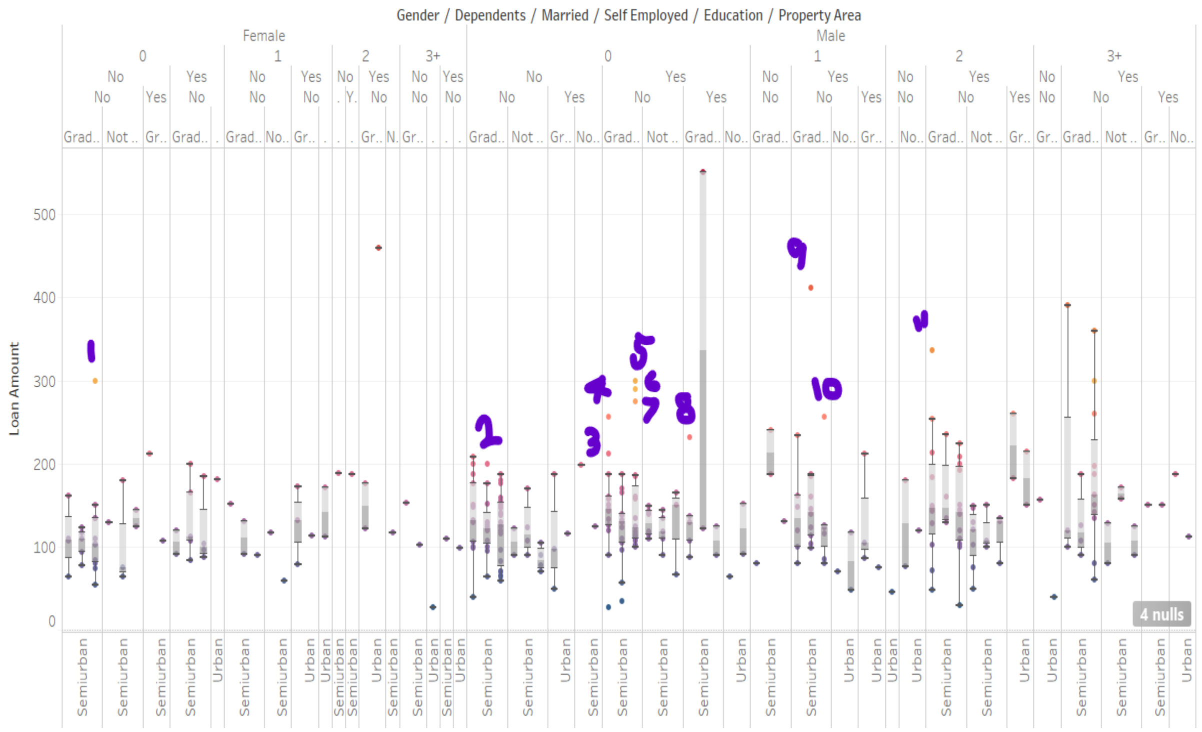
- Somehow a semi-urban male with a credit history of just 2 has been given a loan.
- Applicant income matters more than credit history in the case of urban applicants.
- When comparing credit history with property area, Somehow a rural male with 2 dependents and a credit history of just 6 has gotten a loan approval.
- Other rural males need a credit history of 17 and above for a loan approval.
- A semi urban male with just 4 as credit history has gotten a loan over applicants with similar attributes but a higher credit history.
- Females seem to have an advantage over men in spite of occasionally having a credit history.
- When comparing loan amount with property area, In rural areas credit history doesn’t matter much.
- Semi urban females need just a credit history of 3 for a loan approval.
- Semi urban males with credit history of 13 and above and higher loan amounts get loans.
- Urban females need to have only a credit history of 9 to get a loan but males need only 5 (unknown reasons).

- When comparing loan amount term with property area,10+ null values disrupt inferences.



- When comparing applicant income with self-employment status , Males with credit history of 20 and above get loan approvals when not self employed.
- Self-employed females need just a credit history of 2 and males need a credit history of 4 at least.
- When comparing co-applicant incomes with self-employment status, Applicant’s property area matters in the case of someone who’s not self employed.
- Credit history of at least 2 is necessary for someone who’s self-employed.
- When comparing credit history with self-employment status, Credit history of at least 9 is required for applicant who isn’t self-employed.
- Credit history of at least 3 is required for self-employed applicants.
- When comparing loan amount with self-employment status, Male applicants need a credit history of 16 and above for loan approval if they aren’t self-employed.
- Credit history doesn’t seem to matter in the case of a self-employed applicant.

- When comparing loan amount term with self-employment status,10+ null values hinder results.



OUTLIER

1) Loan amount of 300 is unusual for an unmarried female graduate from an urban area who isn't self-employed and has no dependents. The upper whisker is at 134 and max is 150.
2) Loan amount of 200 is unusual for an unmarried graduate male from a semi-urban area with no dependents and isn't self-employed. Upper whisker and max is at 176.

3) A loan amount of 212 is unusual for a married graduate rural male who isn’t self-employed as 188 is the maximum loan amount.
4) A loan amount of 257 is unusual for a married graduate rural male who isn’t self-employed as 188 is the maximum loan amount.
5) A loan amount of 275 is unusual for a married graduate urban male who isn’t self-employed as 186 is the upper whisker loan amount.
6) A loan amount of 290 is unusual for a married graduate urban male who isn’t self-employed as 186 is the upper whisker loan amount.
7) A loan amount of 300 is unusual for a married graduate urban male who isn’t self-employed as 186 is the upper whisker loan amount.
8) A loan amount of 232 is unusual for a married graduate rural male who is self-employed as 137 is the upper whisker loan amount.
9) A loan amount of 412 is unusual for a married graduate semi-urban male with a dependent who isn’t self-employed as 187 is the upper whisker loan amount.
10) A loan amount of 256 is unusual for a married graduate urban male with a dependent who isn’t

self-employed as 126 is the upper whisker loan amount.

11) A loan amount of 336 is unusual for a married graduate rural male with 2 dependents who isn’t self-employed as 254 is the upper whisker loan amount.

Conclusion:

The main purpose of the project is to classify and analyse the nature of the loan applicants. From a proper analysis of data set and constraints of the banking sector, seven different graphs were generated and visualized. From the graphs, many conclusions have been made and information were inferred such as short-term loan was preferred by majority of the loan applicants and the clients majorly apply loan for debt consolidation.

By properly analyzing positive qualities and constraints, it can be concluded with confidence that the Naïve Bayes model is extremely efficient and gives a better result when compared to other models. It works correctly and fulfills all requirements of bankers and can be connected to many other systems. There were multiple malfunctions in the computers, content errors and fixing of weight in computerized prediction systems.

Future Work:

This paper work can be extended to higher level in future. Predictive model for loans that uses machine learning algorithms, where the results from each graph of the paper can be taken as individual criteria for the machine learning algorithm. Machine learning helps to understand the factors which affect the specific outcomes most. Other models like neutral network and discriminate analysis can be used individually or combined for enhancing reliability and accuracy prediction.

References:

1. Goyal and R. Kaur, “A survey on Ensemble Model for Loan Prediction”, International Journal of Engineering Trends and Applications (IJETA), vol. 3(1), pp. 32-37, 2016.

2. J. Hamid and T. M. Ahmed, “Developing Prediction Model of Loan Risk in Banks using Data Mining”.
3. G. Shaath, “Credit Risk Analysis and Prediction Modelling of Bank Loans Using R”
4. Aboobyda Jafar Hamid and Tarig Mohammed Ahmed, “Developing Prediction Model of Loan Risk in Banks using Data Mining”, Machine Learning and Applications: An International Journal (MLAIJ), Vol.3, No.1, pp. 1-9, March 2016.
5. S. Vimala, K.C. Sharmili, ―Prediction of Loan Risk using NB and Support Vector Machine‖, International Conference on Advancements in Computing Technologies (ICACT 2018), vol. 4, no. 2, pp. 110-113, 2018.
6. Aditi Kacheria, Nidhi Shivakumar, Shreya Sawkar, Archana Gupta, Loan Sanctioning Prediction System‖, International Journal of Soft Computing and Engineering (IJSCE), vol. 6, no. 4, pp. 50-53, 2016.
7. X. Francis Jency, V.P.Sumathi, Janani Shiva Sri, “An Exploratory Data Analysis for Loan Prediction Based on Nature of the Clients”,International

Journal of Recent Technology and Engineering (IJRTE), Vol. 7, No. 48, pp. 176-179, 2018.

8. Anchal Goyal, Ranpreet Kaur,"Loan Prediction Using Ensemble Technique", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 3, pp. 523 – 526, March 2016.
9. Anchal Goyal, Ranpreet Kaur, —Accuracy Prediction for Loan Risk using Machine Learning Models‖, International Journal of Computer Science Trends and Technology (I JCST), Vol. 4, Issue 1, pp. 52-57, Jan - Feb 2016
10. Sudhamathy, —Credit Risk Analysis and Prediction Modelling of Bank Loans using R‖, International Journal of Engineering and Technology (IJET), Vol. 8, No. 5, pp. 1954-1966, Oct-Nov 2016.
11. Pidikiti Supriya, Myneedi Pavani, Nagarapu Saisushma, Namburi Vimala Kumari, K. Vikas, "Loan Prediction by using Machine Learning Models", International Journal of Engineering and Techniques, Vol. 5, Issue 2, pp. 144-148, Mar-Apr 2019.

12. Kumar Arun, Garg Ishan, Kaur Sanmeet, “Loan Approval Prediction based on Machine Learning Approach”, IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 18, Issue 3, pp. 79-81, Ver. I (May-Jun.2016).
13. K. Kavitha, “Clustering Loan Applicants based on Risk Percentage using K-Means Clustering Techniques”, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6(2), pp.162–166, 2016.
14. How Decision tree algorithm works, https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-work
15. Naive Bayes — Probabilistic Algorithm, https://medium.com/datadriveninvestor/naive-bayes-probabilistic-algor
16. Loan Prediction Dataset Source, https://www.kaggle.com/uttam96/
17. Supervised Learning with Python, https://www.datascience.com/blog/
18. https://en.wikipedia.org/wiki/Exploratory_data_analysis
19. https://pandas.pydata.org/pandas-docs/stable/

20. https://www.experian.com/blogs/ask-experian/credit-education/score-basics/what-is-a-good-credit-score/

THANK YOU…